SPELL CHECKING AND CORRECTION FOR ARABIC TEXT RECOGNITION

BY

ADNAN ABDO MOHAMMED MAHDI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

January 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ADNAN ABDO MOHAMMED MAHDI** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**

Thesis Committee

Mahmoud 21/3/2012
Prof. Sabri A. Mahmoud (Chairman)

[illegible] 4/4/2012
Dr. Wasfi Al-Khatib (Member)

[illegible] 21-3-2012
Dr. Husni Al-Muhtaseb (Member)

[illegible]
Dr. Adel F. Ahmed
(Department Chairman)

[illegible]
Dr. Salam A. Zummo
(Dean of Graduate Studies)

8/4/12
Date:

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

DEDICATED TO

I dedicate this dissertation with all of my love to my parents, my wife, my sons, my brothers and sisters.

ACKNOWLEDGMENT

First and foremost thanks to Allah who gave me strength, patience and ability to accomplish this research.

I wish to express my appreciation to **Prof. Sabri A. Mahmoud**, who served as my major advisor, for his guidance and patience through the thesis, his continuous support and encouragement can never be forgotten. Thanks are due to my thesis committee members **Dr. Wasfi Al-Khatib** and **Dr. Husni Al-Muhtaseb** for their cooperation, comments and support. Thanks are also due to the Chairman of Information and Computer Science Department **Dr. Adel Ahmed** for his continuous support.

I would like to thank **King Fahd University of Petroleum & Minerals (KFUPM)** for supporting this research and providing the computing facilities. I also thank **King Abdulaziz City for Science and Technology (KACST)** for supporting this research work under NSTIP project 08-INF99-4 “Automatic Recognition of Handwritten Arabic Text (ARHAT)”.

I also would like to thank **Al- Hodeidah University**, which gave me the opportunity for completing my M.Sc. degree in KFUPM.

I owe thanks to my colleagues and my friends for their help, motivation and pivoted support. A few of them are Rashad Ahmed, Esam Al-Nahari, Mohammed Amro, Mawaal Ali, Hamzah Luqman, Abdullah Qalisi, Dia Abu Zeina, Fadhel Alsheikh, Irfan Ahmed, Mojeeb Al-khiaty, Abdul Mageed Hendi, Mohammed Ali, Ahmed Barada, Omar Shaaban and many others; all of whom I will not be able to name here.

TABLE OF CONTENTS

DEDICATED TO III

ACKNOWLEDGMENT IV

TABLE OF CONTENTS V

LIST OF TABLES VIII

LIST OF FIGURES XI

THESIS ABSTRACT XII

ملـخص الرســالة XIV

CHAPTER 1 INTRODUCTION 1

1.1. INTRODUCTION 1

1.2. PROBLEM STATEMENT 2

1.3. THESIS OBJECTIVES 3

1.4. RESEARCH METHODOLOGY 4

1.5. THESIS OUTLINES 5

CHAPTER 2 BACKGROUND 6

2.1. SPELLING TYPES OF ERRORS 7

2.1.1. Typographic Errors 7

2.1.2. Cognitive Errors 8

2.1.3. Phonetic errors 8

2.1.4. OCR Errors 8

CHAPTER 3 LITERATURE REVIEW 10

3.1. INTRODUCTION 10

3.2. Non-Word Error Detection 10

3.3. Isolated-Word Error Correction 11

3.4. Context-Dependent Word Correction 19

3.5. Arabic Spell Checking and Correction 22

CHAPTER 4 ARABIC SPELL CHECKING, CORRECTION AND LANGUAGE MODEL 27

4.1. Arabic spell checking and correction prototype 27

4.2. Arabic Language Model 32

4.2.1. Corpus 32

4.2.2. Dictionary 38

4.2.3. Statistical Language Models 39

4.2.4. Data Preparation 43

CHAPTER 5 ARABIC SPELL CHECKING PROTOTYPE 47

5.1. Dictionary look-up 47

5.2. Morphological Analyzer 48

5.3. Character n-grams 54

5.4. Experimental results 56

5.4.1. Evaluation Measures 56

5.4.2. Spell Checking Performance 57

5.5. Summary 89

CHAPTER 6 ARABIC SPELL CORRECTION PROTOTYPE 92

6.1. Generating Candidate Words 92

6.2. Ranking Candidate Words 96

6.3. CORRECTING ERRONEOUS WORDS 98

6.4. ARABIC SPELL CORRECTION USING OCR CONFUSION MATRIX.......................... 100

6.4.1. Generating Candidate Words.. 101

6.4.2. Ranking Candidate Words.. 109

6.4.3. Correcting Erroneous Words 109

6.5. EXPERIMENTAL RESULTS 110

6.5.1. Evaluation Measures.. 110

6.5.2. Spell Correction Performance... 112

CHAPTER 7 CONCLUSION AND FUTURE WORK 121

REFERENCES 123

VITA 128

LIST OF TABLES

Table 4.1 Excluded files from CCA 33
Table 4.2 Corpus characteristic 35
Table 4.3 Character N-grams 41
Table 4.4 Data sizes of the corpus (Word N-grams) 42
Table 4.5 Introducing three types of errors 45
Table 5.1 An example of the character frequency bi-gram array 55
Table 5.2 Spell checking results on ATR data using BAMA 58
Table 5.3 Spell checking results on CG1-5 using BAMA 59
Table 5.4 Spell checking results on CG1-10 using BAMA 60
Table 5.5 Spell checking results on CG2-5 using BAMA 61
Table 5.6 Spell checking results on CG2-10 using BAMA 62
Table 5.7 Spell checking results on CG1-5 using Dictionary look-up 63
Table 5.8 Spell Checking results on ATR data using Dictionary look-up after update the Dictionary 63
Table 5.9 Spell checking results on CG1-5 using Dictionary look-up 64
Table 5.10 Spell Checking results on CG1-10 using Dictionary look-up 65
Table 5.11 Spell checking results on CG2-5 using Dictionary look-up 66
Table 5.12 Spell checking results on CG2-10 using Dictionary look-up 67
Table 5.13 Spell checking results on ATR data using a combination of 68
Table 5.14 Spell checking results on CG1-5 using a combination of 69
Table 5.15 Spell checking results on CG1-10 using a combination of 70
Table 5.16 Spell checking results on CG2-5 using a combination of 71
Table 5.17 Spell checking results on CG2-10 using a combination of 72
Table 5.18 Spell checking results on ATR data using character Bi-gram 73
Table 5.19 Spell checking results on ATR data using character Tri-gram 74
Table 5.20 Spell checking results on ATR data using character Quad-gram 75

Table 5.21 Spell checking results on ATR data using a combination of character n-grams 76
Table 5.22 Spell checking results on CG1-5 data using character Bi-gram.................... 77
Table 5.23 Spell checking results on CG1-5 data using character Tri-gram................... 77
Table 5.24 Spell checking results on CG1-5 data using character Quad-gram............... 78
Table 5.25 Spell checking results on CG1-5 data using the combination of character n-grams 79
Table 5.26 Spell checking results on CG1-10 data using character Bi-gram.................. 80
Table 5.27 Spell checking results on CG1-10 data using character Tri-gram................. 81
Table 5.28 Spell checking results on CG1-10 data using character Quad-gram............. 81
Table 5.29 Spell checking results on CG1-10 data using the combination of character n-grams 82
Table 5.30 Spell checking results on CG2-5 data using character Bi-gram.................... 83
Table 5.31 Spell checking results on CG2-5 data using character Tri-gram.................. 84
Table 5.32 Spell checking results on CG2-5 data using character Quad-gram............... 85
Table 5.33 Spell checking results on CG2-5 data using the combination of character n-grams 85
Table 5.34 Spell checking results on CG2-10 data using character Bi-gram.................. 86
Table 5.35 Spell checking results on CG2-10 data using character Tri-gram................. 87
Table 5.36 Spell checking results on CG2-10 data using character Quad-gram............. 88
Table 5.37 Spell checking results on CG2-10 data using the combination of character n-grams 88
Table 5.38 Spell checking results on Arabic Text Recognition (ATR) data................... 89
Table 5.39 Spell checking results on CG1-5 data 90
Table 5.40 Spell Checking results on CG1-10 data 90
Table 5.41 Spell Checking results on CG2-5 data 91
Table 5.42 Spell Checking results on CG2-10 data 91
Table 6.1 List of candidate words from the misspelled word "والمجنمع"........................... 96
Table 6.2 The characters insertion counts 104

Table 6.3 The characters deletion counts 105
Table 6.4 The characters substitution counts 106
Table 6.5 Spell Correction Results on ATR Data using OCR Confusion Matrix......... 114
Table 6.6 Spell correction results on ATR data using Edit Distance 116
Table 6.7 Spell correction results on CG1-5 data... 117
Table 6.8 Spell correction results on CG1-10 data.. 118
Table 6.9 Spell correction results on CG2-5 data.. 119
Table 6.10 Spell correction results on CG2-10 data.. 120

LIST OF FIGURES

Figure 3.1 Proposals and morphological analysis for the misspelling bos (Agirre et al. 1998) 15

Figure 3.2 Only valid proposals are kept (Agirre et al. 1998) 15

Figure 4.1 Arabic Spell Checking and Correction prototype 30

Figure 4.2 Implementation structure of Arabic Spell Checking and Correction 31

Figure 4.3 An example of text distorted 33

Figure 4.4 A sample of an article from the Al-Hayat corpus 37

Figure 4.5 Examples of misspelled words in the corpus resulting from typos 37

Figure 4.6 General scheme to convert corpus to dictionary 38

Figure 4.7 Sample language model generated by IRSTLM 42

Figure 4.8 Data preparation 43

Figure 4.9 Algorithm for introducing one of three different types of errors 46

Figure 5.1 Illustration of the BAMA algorithm when the word has both the prefix and the stem 52

Figure 5.2 Illustration of the BAMA algorithm when the word has both the stem and the suffix 52

Figure 5.3 Illustration of the BAMA algorithm when the word has the prefix, the stem and the suffix 53

Figure 6.1 Flowchart for generating candidate words based on the minimum edit distance 94

Figure 6.2 Illustration of the spell correction process 99

Figure 6.3 OCR confusion matrix 108

Figure 6.4 An example of the probability of the candidate word "الحلاوة" is less than other candidate's word 113

Figure 6.5 List of un-corrected OCR errors using OCR confusion matrix 114

Figure 6.6 List of un-corrected OCR errors using Edit Distance 115

Figure 6.7 Error correction accuracy 116

THESIS ABSTRACT

Name:	Adnan Abdo Mohammed Mahdi
Title:	Spell Checking and Correction for Arabic Text Recognition
Major Field:	Computer Science
Date of Degree:	January 2012

The problem of automatic spell checking and correction is one of the active research problems in the area of Natural Language Processing (NLP). The importance of spell checking and correction originates from the fact that they are useful in many fields such as, word processing, information retrieval, grammar correction and machine translation. In addition, they are important in correcting errors in OCR.

In this thesis, a successful spell checking and correction prototype for Arabic text is designed and developed. This work consists of collecting Arabic text corpus from different topics such as news, short stories, and books. Several types of language models (N-gram and dictionary) are used.

For spell checking, we used Buckwalter's Arabic morphological analyzer (BAMA), a dictionary look-up and a language model on character level (character *n*-grams). For spell correction, we used edit distance technique, N-grams language models (word *n*-grams) and OCR confusion matrix.

In order to test our spell checking and correction prototype, two types of data sets are used. The first set, Arabic Text Recognition (ATR) data, which is generated from an OCR system developed at KFUPM. The second set, a Computer Generated (CG) data, which is prepared by taking a normal correct text and randomly introducing three types of errors (*insert, delete and replace*).

The accuracy results of spell checking techniques are compared in terms of recall, precision and F_1-measure. The results of combining the two techniques (viz. Buckwalter's Arabic morphological analyzer, dictionary look-up and language model on character level) are presented and analyzed. The best method is the one which combine the Buckwalter's Arabic morphological analyzer (BAMA) and the dictionary look-up.

The accuracy results of spell correction techniques are presented and analyzed (viz. edit distance, language model on word level and OCR confusion matrix). The results show that the edit distance and language model techniques give good results on the Arabic Text Recognition (ATR) data and Computer Generated (CG) data.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN, SAUDI ARABIA

JANUARY 2012

ملخص الرســالة

الاســــــــم: عدنان عبده محمد مهدي

عنوان الرسالة: التدقيق والتصحيح الإملائي للنص العربي المتعرف عليه آليا

التخصــــص: علوم الحاسب الآلي

تاريخ التخـرج: يناير 2012

تعد مشكلة التدقيق والتصحيح الإملائي واحدة من المشاكل النشطة بحثيا في مجال معالجة اللغة الطبيعية . وتعود هذه الأهمية إلى حقيقة إنها مفيدة في مجالات عدة مثل معالجة النصوص، إسترجاع المعلومات، تصحيح القواعد النحوية والترجمة الآلية. هذا بالإضافة الى أهميتها في تصحيح أخطاء التعرف الضوئي على الحروف (OCR).

في هذه الأطروحة تم تصميم وتطوير نموذج التدقيق والتصحيح الإملائي للنص العربي. ويتكون هذا العمل من مكنز نص عربي مجمع من موضوعات مختلفة مثل الأخبار والقصص القصيرة والكتب. وباستخدام عدة أنواع من النماذج اللغوية (ان-غرام والقاموس). استخدمنا المحلل الصرفي العربي لـ Buckwalter (BAMA)، والبحث في القاموس Dictionary Look-up ونماذج اللغة على مستوى الحرف character n-grams للتدقيق الإملائي. و قد استخدمنا تقنية تحرير المسافة edit distance، نماذج اللغة على مستوى الكلمة word n-grams و مصفوفة الإلتباس (OCR confusion) للتصحيح الإملائي.

ولإختبار نموذجنا للتدقيق الإملائي والتصحيح، فقد قمنا باستخدام مجموعتين من البيانات. المجموعة الأولى، بيانات النص العربي المتعرف عليه آليا، والتي تم توليدها من نظام التعرف الضوئي على الكتابة العربية و الذي تم تطويره في جامعة الملك فهد للبترول والمعادن. والمجموعة الثانية، بيانات تم توليدها بواسطة الحاسوب وقد تم إعدادها بأخذ نص عادي صحيح ، و إدراج ثلاثة أنواع من الأخطاء عشوائيا وهي الإدراج أو الحذف أو الإستبدال في النص.

وقد تمت مقارنة دقة نتائج تقنيات التدقيق الإملائي من حيث الشمولية (recall) والدقة (precision) و قياس ف 1 (F1-measure) وتم جمع نتائج تقنيات المحلل الصرفي العربي لـ Buckwalter والبحث في القاموس ونموذج

اللغة على مستوى الحرف وتم عرضها وتحليلها. وقد تم الحصول على أفضل النتائج بجمع المحلل الصرفي العربي لـ Buckwalter (BAMA) والبحث في القاموس Dictionary Look-up.

كما تم عرض وتحليل دقة نتائج تقنيات تحرير المسافة ونماذج اللغة على مستوى الكلمة word n-grams و مصفوفة الإلتباس (OCR confusion) للتصحيح الإملائي. وقد أظهرت النتائج إن إستخدام تحرير المسافة مع تقنيات نماذج اللغة أعطى نتائج جيدة على كل من بيانات النص العربي المتعرف عليه آليا والبيانات المولدة بواسطة الحاسوب.

ماجستير العلوم

جامعـــة الملك فهـــد للبترول والمعـــادن

الظهران – المملكة العربية السعودية

يناير 2012

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION

The problem of automatic spell checking and correction is one of the active researched problems in the area of Natural Language Processing (NLP). It has a long tradition in the history of computer technology (Damerau 1964) (Riseman & Hanson 1974) (Kukich 1992). The research started as early as 1960s (Damerau 1964) and it has continued up to the present. Spell checking and correction have proved their usefulness in various applications such as word processing, information retrieval, grammar correction, machine translation, optical character recognition (OCR), etc. (Kukich 1992). The proficiency of misspelling word correction can improve the efficiency of those applications. During the last 40 years, many techniques for detection and correction of spelling errors were proposed, such as, dictionary look up techniques, N-gram models, minimum edit distance, similarity key techniques, probabilistic and rule based techniques. These techniques are often designed based on the study of spelling error patterns.

Most researchers classify spelling errors into two main groups:(1) non-word errors, words that have no meaning and do not exist in a lexicon or a dictionary such as "houe" for "house" and (2) real-word errors, dictionary words that are less likely to be used in the given context such as mistaking “their” with “there”. Another categorization was given by Karen Kukich (Kukich 1992), who divided spelling errors into three types: (1) cognitive errors, for example, the word “receive” is often mistakenly written as “recieve”, (2) phonetic errors, such as, writing “nacherly” instead of “naturally” which are both phonetically correct and (3) typographical errors such as writing “teh” instead of “the”. In this thesis, we are proposing a prototype for spell checking and correction for Arabic text recognition that would be able to detect and correct non-word errors automatically. Dictionary lookup technique, Buckwalter’s Arabic morphological analyzer and N-grams language models on the character level are used to detect spelling errors. Edit distance techniques, N-grams language models and OCR confusion matrix are used to perform spell correction.

1.2. PROBLEM STATEMENT

A spell checker is a necessary element for detecting and correcting spelling errors in a text in any language. The detection and correction of spelling errors have important roles in modern word processors. They are also important in correcting errors of OCR output and on-line handwriting recognition. Based on that importance, extensive work has been done in the area for English and other languages. However, Arabic has not received similar interest. The problem in Arabic language is the absence of a general system for detecting and correcting Arabic spelling errors. Moreover, there is a lack of an automatic spelling corrector without the need for human interference, and without wasting much efforts and time when correcting in the traditional method.

1.3. THESIS OBJECTIVES

The main objective of this research is to design and implement a prototype for spell checking and correction for Arabic text recognition that would be able to detect and correct non-word errors automatically. In order to accomplish this objective, the following tasks have to be performed:

1. Perform a literature survey of spell checking and correction.
2. Identify the spell checking and correction techniques.
3. Identify and collect suitable Arabic text corpus.
4. Analyze the corpus and build suitable language models for spell checking and correction (N-grams, dictionary).
5. Analyze spelling errors made by Arabic Optical Character Recognition system.
6. Implement an Arabic spell checking and correction prototype.
7. Evaluate the performance of the proposed spell checking and correction prototype.
8. Identify factors that can improve the performance of the proposed spell checking and correction system.
9. Analyze the results and present the conclusions.

1.4. RESEARCH METHODOLOGY

In order to achieve the aforementioned objectives, this research has used a methodology encompassing the following phases:

- ***Phase 1: Literature Review***

  Perform a literature review on the problems of detecting and correcting spelling errors and the most important techniques that address each of these problems.

- ***Phase 2: Data collection and generation***

  In this phase, we collect and generate the following:

  - Develop a corpus from a large collection of Arabic text spanning different subjects.
  - Build a dictionary from the corpus.
  - Generate the confusion list of characters.
  - Generate the statistics of N-grams (uni-grams, bi-grams and tri-grams).
  - Use OCR data in the generation of the confusion list and for testing the prototype.

- ***Phase 3: Implement the proposed prototype for Arabic Spell Checking and Correction***

In this phase, we implement the proposed prototype for Arabic spell checking and correction.

- ***Phase 3.1: Spell checking Module***

Implement the spell checking module, using a dictionary look-up technique, a morphological analyzer, and character N-grams.

- ***Phase 3.2: Spell Correction Module***

Implement the spell correction module using edit distance, word N-grams and OCR confusion matrix.

- ***Phase 4: Experimental results***

The experimental results of Arabic spell checking and correction on OCR and Computer Generated data are addressed.

1.5. THESIS OUTLINES

The remainder of this thesis is organized as follows. Chapter 2 presents basic terminology and background information on spell checking and correction. We review the related work in Chapter 3. Chapter 4 presents a prototype for Arabic spell checking and correction. Chapter 5 presents Arabic spell checking while Chapter 6 presents Arabic spell correction. Finally, conclusions and future direction are addressed in Chapter 7.

CHAPTER 2

BACKGROUND

This chapter addresses spell checkers classification and the process of any spell checking.

Spell checkers can be classified into two types: interactive and automatic (Naseem 2004) (Verberne 2002). In the interactive spell checker, the spell checker detects misspelled words. It then suggests more than one possible correction for each misspelled word and allows the user to choose one of the suggested corrections. In automatic spell checker, the misspelled word is automatically replaced with the most probable word without interaction with the user.

The spell checking process can be divided into three steps (Naseem & Hussain 2007) (Verberne 2002) : (1) error detection; (2) finding candidate correction words and (3) ranking candidate words. Error detection refers to the process of finding misspelled words in a text while finding candidate correction words refers to the process of finding the suggested corrections. Ranking refers to the process of ranking the suggested corrections in decreasing order of probability.

2.1. SPELLING TYPES OF ERRORS

Many studies were performed to analyze the types of spelling errors (Damerau 1964) (Kukich 1992). Damerau (Damerau 1964) found that approximately 80% of all misspelled words (non-word) are single-error misspellings. So, according to these studies, spelling types of errors can result from human typed errors and text recognition errors like OCR or handwriting recognition. The number and nature of spelling errors are different. Spelling types of errors caused by human can be classified into three groups, typographic errors, cognitive errors and phonetic errors (Kukich 1992) (Haddad & Yaseen 2007).

2.1.1. Typographic Errors

Typographic errors occur when a writer knows how to spell the word but makes a mistake when writing the word (Bandyopadhyay 2000). These errors are related to the keyboard adjacencies. The typographic errors belong to one of the following (Damerau 1964):

1. Letter insertion, e.g. typing "العللوم" for "العلوم", the letter (/ ل/) is additionally inserted.
2. Letter deletion, e.g. typing "متبة" for "مكتبة", the letter (/ ك/) is deleted.
3. Letter substitution, e.g. typing "التص" for "النص", the letter (/ن/) is mistakenly substituted by (/ت/).

4. Transposition of two adjacent letters, e.g. typing "اجمتاع" for "اجتماع", the letters (/ت/) and (/م/) are swapped.

2.1.2. Cognitive Errors

Cognitive errors occur when a writer does not know how to spell the word due to lack of knowledge (Bandyopadhyay 2000), for example, “لكن” by “لاكن”

2.1.3. Phonetic errors

Cognitive errors include phonetic errors where a word has been replaced by similar sounding word, for example, "قضاة" by "قضات" or "عظيم" by "عضيم".

2.1.4. OCR Errors

OCR errors occur from OCR misrecognized text of the original document. English text errors are grouped as follows:

- Substitutions, for example *c* → *e ;*
- Multi substitutions (single characters recognized as multiple characters, for example *m* → *rn* or *n* → *ii* or multiple characters recognized as one character, for example *cl* → *d* or *iii* → *m* ;
- Space insertion, for example *cat* → *c at* ;
- Space deletion like the cat → thecat

- Letter insertion, for example write → writte
- Letter deletion, for example house → huse

However, Arabic text errors are grouped as follows:

- Substitutions, for example *ص→ض [المفاصل→المفاضل] or ظ →ط [ظليلة→طليلة]* ;
- Multi substitutions (single characters recognized as multiple characters), for example *س→ مص [ليس→اليمص]* ;
- Space insertion, for example يذهب → يذ هب;
- Letter insertion, for example *ببعض* → *ببعضل* or *زيت* → *زيدت*;
- Letter deletion, for example *الذي* → *لذي* or *القروح* → *اقروح or طيبة* → *طية*

CHAPTER 3

LITERATURE REVIEW

3.1. INTRODUCTION

The problem of automatic spell checking and correction has been studied for decades (Kukich 1992). It has become a perennial research challenge. Work on computer techniques for automatic spelling error detection and correction in text started in the 1960s (Damerau 1964).

A number of researchers have carried out extensive work and published papers on spell checking and correction (Kukich 1992) (Bowden & Kiraz 1995) (Liang 2008). Most of the published works focused on three main issues: (1) non-word error detection; (2) isolated-word error correction; and (3) context-dependent word correction (Kukich 1992). There are many techniques discussed to tackle these problems. Lorraine Liang (Liang 2008) in her master thesis, Karen Kukich (Kukich 1992) in her comprehensive survey and (Deorowicz & Ciura 2005) discussed the most important techniques that address each of these problems.

3.2. NON-WORD ERROR DETECTION

A non-word can be defined as a sequence of letters that is not a valid word in the language in any context (Boyd 2009). That means a non-word error results from nonexistent word in the language. Work on non-word error detection started form the early 1970s (Kukich 1992). The techniques that have been explored for non-word error detection can be divided into two main categories: dictionary lookup techniques and n-gram analysis (Kukich 1992).

Zamora et al. (E. Zamora et al. 1981) presented a study to evaluate the effectiveness of tri-gram frequency statistics for detecting spelling errors, verifying correctly-spelled words, locating the error position within a misspelling, and distinguishing between the basic kinds of spelling errors. Their study was applied to 50,000 word/misspelling pairs collected from seven chemical abstract service databases. The tri-gram frequency table was compiled from a large corpus of text. They found that the tri-gram analysis technique was able to determine the error location within a misspelled word accurately within one character 94% of the time.

After a non-word has been detected, one or more words need to be selected as candidate corrections. So, the most common method used to correct non-word errors is the isolated-word error correction.

3.3. Isolated-Word Error Correction

Isolated-word error correction refers to spell correction without taking any context or linguistic information in which the misspelling occurs. Work on isolated-word error correction started as early as the 1960s into the present. During that time, many different techniques have been devised, such as minimum edit distance (Brill & Moore 2000)

(Magdy & Darwish 2006) (Magdy & Darwish 2010), similarity key techniques (Zobel et al. 1996), n-gram based techniques (Riseman & Hanson 1974) (Islam & Inkpen 2009a) (Islam & Inkpen 2009b), probabilistic techniques (Kemighan et al. 1990) (Church & Gale 1991), rule-based techniques (Yannakoudakis & Fawthro 1983) (Shaalan et al. 2003) and phonetic similarity techniques (Schaback & F. Li 2007). These techniques are used to correct non-word errors.

Yannakoudakis and Fawthrop (Yannakoudakis & Fawthro 1983) proposed a rule-based system to correct spelling mistakes. They used a set of rules that describe common spelling mistakes, such as singling and doubling consonants, and an expert system to traverse those rules and transform the misspelled word according to the rules. After every transformation, the word is looked up in the dictionary searching for a match and a list of candidates is generated. In addition, they gave every rule a certain probability depending on its frequency in a corpus. For instance, they counted the frequency a consonant was mistakenly doubled in the corpus and they ordered the candidates suggestions according to predefined estimates of the probabilities of the rules that generated them. One thing to mention here is that the rules only generate words that are one or two errors different from the original word. In addition, visiting the dictionary after the application of each rule to match the input word is costly.

Angell et al. (Angell et al. 1983) developed a technique based on the use of tri-grams for spelling correction applications for English text. Their technique computed a similarity measure based on the number of tri-grams that occurred in both a misspelled word and a dictionary word. The similarity measure was then computed by a simple function called Dice coefficient (2*(c/(n+n')) where c is the number of common tri-grams for both the misspelled word and the word in the dictionary, n is the length of misspelled word, and n' is the length of the dictionary word). The drawback of this technique is that any words shorter than three characters cannot be detected. They tested their technique on a test set of 1,544 misspelled words using a dictionary of 64,636 words and reported an overall accuracy score of 76%.

Kemighan et al. (Kemighan et al. 1990) and Church and Gale (Church & Gale 1991) devised an algorithm, called CORRECT, to propose a list of candidate corrections for English language. Their approach was based on the noisy channel model. They used minimum edit distance technique to generate a set of candidate corrections that differ from the misspelled word by a single insertion, deletion, substitution or transposition. Then, a Bayesian formula was used to rank the candidate suggestions. Their proposed method achieved 87% correction rate. Their work is limited to the correction of words with a single typographical error.

Elmi and Evens (Elmi & Evens 1998) presented a framework to correct spelling mistakes based mainly on the edit distance. They used a lexicon of around 4500 words and they divided it into 314 different segments according to the starting letters and the sizes of the words. Then they calculated the edit distance of the suspected word from the words in the lexicon and they assigned a weight to each distance. If the error appears at the first letter then the weight is increased by 10%. On the other hand, if the error is due to substituting a character with its neighbor in the keyboard then the weight is reduced by 10%. Moreover, they identified four categories of errors: reversed order, missing character, added character and substituted character. In order to identify the reversed order and character substitution errors, they checked the part of the lexicon that starts with the same letter and has the same word size. As for the missing character and added character errors, they checked the parts of the lexicon that start with the same letter and has the word size + 1 or - 1 respectively. After that, they come up with a candidates list depending on a fixed threshold for the edit distance. Finally, they filtered and ordered the candidate list by applying semantic rules such as the tense of the verb with respect to the rest of the sentence using a part-of-speech tagger.

Agirre et al. (Agirre et al. 1998) presented a system to correct non-words that are not found in a dictionary. After identifying non-words, they used a part-of-speech tagger to tag the input text. Then they used a morphological analyzer to generate all morphological interpretations of the misspelled word. Figure 3.1 shows an example of a misspelled word (viz. “bos”). Then they used linguistic constraints and rules to eliminate wrong possibilities. After that, they filtered the results by finding the probability of occurrence of every suggestion in a certain corpus and then they returned the most probable candidate. For instance, a plural pronoun (PRON PL) cannot be followed by a singular verb (V S) so the word “bop” is discarded, as shown in Figure 3.2.

<our>
"our" PRON PL…
<bos>; SPELLING ERROR
"boss" N S
"boys" N P
"bop" V S
"Bose" <Proper>

Figure 3.1 Proposals and morphological analysis for the misspelling bos (Agirre et al. 1998)

<our>
"our" PRON PL…
<bos>; SPELLING ERROR
"boss" N S
"boys" N P
~~"bop" V S~~
"Bose" <Proper>

Figure 3.2 Only valid proposals are kept (Agirre et al. 1998)

Brill and Moore (Brill & Moore 2000) proposed an improved model for spelling correction using the noisy channel model and Bayes' rule. Their model used dynamic programming for finding edit distance between a misspelled word and a dictionary word. A 10,000 word corpus of common English spelling errors, paired with their correct spelling is used. Different context window sizes to evaluate their model were used. Then for each size, they calculated the percentage of time the correct word was in the best three, best two or best one word candidates. They reported that their model has an accuracy of 93.6%, 97.4% and 98.5% in being the best one, two and three word candidates, respectively.

Taghva and Stofsky (Taghva & Stofsky 2001) proposed an interactive system for correcting spelling mistakes induced by OCRs. The system is called OCRSpell and was used as a post-processor of the scanned OCR documents. The system used confusion sets that included the most common mistakes made by OCRs (e.g. rn → m and ii→n). These sets were used to identify misspelled words. The system was composed of five models:

1. A parser designed specifically for OCR-generated text;
2. A virtual set of domain specific lexicons;
3. The candidate word generator;
4. The global/local training routines (confusion generators);
5. The graphical user interface.

After generating the list of candidates, which are ranked according to their probabilities, the system lets the user make the correction. One disadvantage of this system is that it does not automatically correct or suggest candidates for errors generated by merging two words or separating a single word and it lets the user handle these types of mistakes.

Hodge and Austin (Hodge & Austin 2002) compared standard spell checking algorithms to a spell checking system based on a modular binary neural network architecture (AURA) that uses correlation matrix memories (CMMs) (Austin 1996), (Turner & Austin 1997). They proposed a simple spell checker using efficient associative matching in the AURA modular neural system. Their approach aimed to provide a pre-processor for an information retrieval (IR) system that allows the user's query to be checked against a lexicon. Then, any spelling errors are corrected, to prevent wasted searching time. They used an integrated hybrid approach, n-gram approach and Hamming Distance, to overcome the four main forms of spelling errors: insertion, deletion, substitution and transposition.

Tahira Naseem (Naseem 2004) proposed a hybrid approach for Urdu spelling error correction. Her approach was based on single edit distance technique to correct typographic errors and Soundex[1] to correct phonetics errors. She tested her approach on a test data of 744 errors, which included 444 space related errors and 280 non-space errors using dictionary of around 112,481 words. An overall accuracy score of 96.68% was reported.

Lehal (Lehal 2007) presented the complete design and implementation of a Punjabi spell checker. His system was designed to detect and correct non-word errors. The first step in his work was the creation of a lexicon of correctly spelled words in order to check the spellings as well as generate the suggestions. He stored all the possible forms of words of Punjabi lexicon. After that, he partitioned the lexicon into sixteen sub-dictionaries based on the word length. He used dictionary lookup to detect misspelled words. After identifying the misspelled words, he used reverse minimum edit distance between a misspelled word and a dictionary word to generate list of candidate

[1] http://en.wikipedia.org/wiki/Soundex

words. In addition, he added words, which are phonetically similar to the misspelled words to the suggestion list. After that, he sorted the suggestion list based on the phonetic similarity between the suggested word and the misspelled word, the frequency of occurrence of the suggested word, and the smallest number of substitutions, insertions and deletions required to change the misspelled word to the suggested word. He tested his spell checker on a test set of 255 misspelled words. The percentage of occurrence of the correct words at the top of the suggestion list was 81.14% and in the top 10 words was 93.4%. One drawback of the method is using small testing data, only 255 misspelled words.

Kaur and Bhatia (Kaur & Bhatia 2010) discussed the design, techniques and implementation of the developed spell checker for Punjabi Language. Their system, SUDHAAR, was designed for spell checking of Punjabi text. Their system was mainly designed to detect and correct non-word errors. The system was composed of three modules viz. Creation of Punjabi Dictionary, Error Detection and Error correction and Replacement. They used Creation of Punjabi Dictionary module to build a corpus which contains around one million unique correct Punjabi words. They used dictionary look up technique to detect the errors in the input text. Then they used error pattern analysis (Bhagat 2007), and applied Minimum Edit Distance to find suitable suggestions which were added to suggestion list. Finally, the system allowed the user to select the word from the suggestion list. The system was reported to detect approximately 80% of the errors and provides 85% of the correct suggestions.

3.4. Context-Dependent Word Correction

Context-dependent word correction refers to spell correction that would correct errors involving textual and linguistic context (Liang 2008). Work on the context-dependent word correction began in the early 1980s with the development of automatic natural language processing models.

Golding (Golding 1995) suggested an approach for context-sensitive spell checking. In his approach, he used confusion sets to identify potential spelling mistakes. For instance, the word “weather” is often confused with the word “whether” and vice versa so the context in which they are used is considered to resolve the ambiguity. The confusion sets used in his work were taken from the list of “Words Commonly Confused" in the back of the Random House Unabridged dictionary (Flexner 1983). The approach was tested using two classifiers viz. The decision lists and the Bayesian classifier. This technique showed a high performance rate. It depended heavily on the selected confusion sets and the corpus used to collect the features.

Golding and Schabes (Golding & Schabes 1996) devised a hybrid method, *Tribayes*, that combined a part-of-speech tri-gram method with a Bayesian hybrid method to detect and correct real word errors. They used the confusion sets from the list of Words Commonly Confused in the back of the Random House Unabridged Dictionary (Flexner 1983). The tri-gram method was used to make decisions using the confusion sets while the Bayesian method was used to predict the correct word. Given a target occurrence of a word to correct, *Tribayes* substituted in turn each word from the confusion set into the sentence. For each substitution, it calculated the probability of the resulting sentence based on part-of-speech tri-grams. It selected as its suggestion the word that yields the sentence having the highest probability of all confusion sentences. The disadvantage of the method is that new confusable errors will never be corrected.

Fossati and Eugenio (Fossati & Eugenio 2007) proposed a method of mixed tri-grams model that combines the word tri-grams model and POS-tri-gram model. They defined confusion sets for all words in the vocabulary using minimum edit distance less or equal to 2. One of the limitations of their approach is the lack of using a good smoothing technique for assigning probabilities of unseen tri-grams. Another limitation is using small training data. In addition, it skips words with less than three characters.

Islam and Inkpen (Islam & Inkpen 2009a) presented a method for detecting and correcting multiple real-word spelling errors. Their method focused mainly on how to improve the detection and the correction recall, while maintaining the respective precisions (the fraction of detections or amendments that are correct) as high as possible. They used 3-gram data set from the Google Web 1T corpus[2], which contains English word n-grams (from uni-grams to 5-grams). They presented a normalized and modified version of the Longest Common Subsequence (LCS) string matching algorithm that

[2] www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt

(Islam & Inkpen 2008) (Kondrak 2005) (Melamed 1999) used with different normalization to determine candidate words. Then they used the normalized frequency value of each candidate word as the frequency of the tri-gram containing word over the maximum frequency among all the candidate words. Their method tried to detect and correct for all words except the first word in the input sentence. Their proposed method reported a 89% of detection and a 76% of correction.

Islam and Inkpen (Islam & Inkpen 2009b) further proposed a method for correcting real-word spelling errors using the Google Web 1T n-gram corpus and a normalized and modified version of the Longest Common Subsequence (LCS) string matching algorithm. Their method focused on how to improve the correction recall, while keeping the correction precision (the fraction of suggestions that are correct) as high as possible. They used the same string similarity measure that they used in (Islam & Inkpen 2009a) (Islam & Inkpen 2008) with different normalization to give better similarity value. To determine candidate words of the word having spelling error, they tried to find all the n-grams. First, they used Google 5-gram data set. Then, if the 5-gram data set fails to generate at least one candidate word, they used 4-gram or 3-gram or 2-gram data set. They reported a 88% of correction while maintaining the precision at 91%.

3.5. ARABIC SPELL CHECKING AND CORRECTION

There are few academic papers treating the problem of spell checking and correction for the Arabic language (Shaalan et al. 2003) (Haddad & Yaseen 2007). Recently, spell checkers designed for Arabic language have received great attention due to the increasing Arabic applications that requires spell checking and correction facilities. An example of Arabic spell checkers that are now available for word processing applications is Microsoft word spell checker**.** Most of the Arabic spell checkers are based on simple morphological analysis considering the keyboard effect for correcting single-error misspellings (Haddad & Yaseen 2007)

Shaalan (Shaalan et al. 2003) applied a set of rules for non-words (i.e. words that are not in the dictionary) in Arabic text to correct spelling mistakes. They did not rank or order the list of candidates. The rules they used are as follow:

1- Add missing character: for example candidates for “معض” can be "معوض", "معرض", or "معضد";
2- Replace incorrect character: for example candidates for "معض" can be "كعض", "نعض" or "معد";
3- Remove excessive characters: candidates for "معض" are "مع" and "عض";
4- Add a space to split words: "معض" is split to become "مع" and "عض";

Haddad and Yaseen (Haddad & Yaseen 2007) presented a hybrid model for spell checking and correcting of Arabic words based on semi-isolated word recognition and correction techniques. They considered the morphological characteristics of Arabic scripts in the context of morpho-syntactical, morpho-graphemic and phonetic n-gram binary rules.

There are some hybrid efforts for integrating n-gram and morpho-syntactical analysis in spell checking (Bowden & Kiraz 1995). Although there are some efforts to describe the problems and issues involved in context-dependent spell checking for Arabic, however it is still at an early stage of development.

Zribi et al. (Zribi et al. 2007) proposed a system for the detection and correction of semantic hidden errors occurring in Arabic texts based on Multi–agent System. To detect semantic hidden errors, they checked the semantic validity of each word in the text by combining four methods, Co-occurrence-Collocation, Context-Vector Method, Vocabulary-Vector Method and Latent Semantic Analysis Method. Their system was based on the assumption that there is only one error at most per sentence. To correct semantic errors they generated all the suggested words that were one editing error close to the misspelled word. Then, they substituted the incorrect word with each suggested correction to create a set of candidate sentences. After that, they eliminated all sentences containing semantic anomalies by the detection part of the system. Then, they sorted the remaining sentences using the combined three criteria of classification, typographical distance criterion, proximity value criterion and position of error criterion. They tested their system on 50 hidden errors in 100 sentences and reported 97.05% of accuracy. The limitation of their approach is using a small testing data.

Ahmed Hassan et al. (A. Hassan et al. 2008) proposed an approach for automatic correction of spelling mistakes. Their approach used techniques from finite state theory to detect misspelled words. They assumed that the dictionary is represented as a deterministic finite state automaton. Initially, they build a finite state machine (FSM) that contains a path for each word in the input string. Then they calculated the difference between this FSM and dictionary FSM. This resulted in an FSM with a path for each misspelled word. They created Levenshtein-transducer to generate a set of candidate corrections that have edit distances of 1 and 2 from the input word. In addition, they used confusion matrix to reduce the number of candidate corrections. They used a language model to assign a score to each candidate correction and selected the best scoring correction. They tested their approach using a set of test data composed of 556 misspelled words of edit distances of 1 and 2 on both Arabic and English text and reported accuracy of 89%. The disadvantage of using this approach is that the finite-state transducers (FST) composition process needs long processing time to detect and correct misspelled word.

Magdy and Darwish (Magdy & Darwish 2010) proposed a technique for correction of OCR degraded text that is independent of character-level OCR errors. Their proposed approach did not require the training of a character error model. Instead, they used Levenshtein edit distance with uniform probability distribution for different edit operations. They used a dictionary to check the OCR'ed word and then generated the candidate corrections that are similar to the OCR'ed word. Then, they ranked all candidate corrections by using the following similarity function S_{ED}:

$$S_{ED}(w_i) = \underbrace{e^{-C.ED(w_{OCR}\ ,\ w_i)}}_{Character\ Model} \cdot \underbrace{P(w_i)}_{Unigram\ Model}$$

Where ED is edit distance between OCR word (W_{OCR}) and dictionary word (w_i), while P(w_i) is the uni-gram probability of w_i in the dictionary, and C is relative to the effect of edit distance. The edit distance and tri-gram language model to select the best 5 and 10 candidate corrections are used in their work. The different forms of alef (hamza, alef maad, alef with hamza on top, hamza on wa, alef with hamza on the bottom, and hamza on ya) to alef, and ya and alef maqsoura to ya are normalized. Their proposed technique yielded lower correction effectiveness and required the training of a good language model matching type and style.

Al-Muhtaseb (Al-Muhtaseb 2010) suggested a flexible post-processing stage for correcting the errors of an Arabic OCR System. His proposed approach was composed of four stages: shape to code mapping, error detection, error correction and corrected Arabic text. The error detection module was based on a dictionary related to the used text domain to detect misspelled words. The error correction module was based on the learned knowledge from the analysis of the Arabic OCR. The error correction module was used to tackle substitution, insertion, and deletion errors for every misspelled word. The error correction process was based on the following order: substitution correction, insertion correction and deletion correction. To be noted here, the correction process was stopping when the first correction process results in a correct word. Moreover, this approach was based on the assumption that there is only one error at most in any incorrect word.

Shaalan et al. (Shaalan et al. 2010) proposed an approach for detecting and correcting spelling errors made by non-native Arabic learners. They used Buckwalter's Arabic morphological analyzer to detect spelling errors. Then, the edit distance techniques in conjunction with rule-based transformation approach to correct the misspelled word. They applied edit distance algorithm to generate all possible corrections. Transformation rules to convert a misspelled Arabic word into a possible word correction were applied. Their rules were based on common spelling errors made by Arabic learners. Then, they applied a multiple filtering mechanism (Morphological Analyzer Filter and Gloss Filter) to reduce the generated correction word list. They evaluated their approach using a set of test data composed of 190 misspelled words. Finally, they calculated precision and recall rates for both spelling error detection and correction to measure the performance. An 80+% recall and a 90+% precision were reported. Their test data was designed only to cover common errors made by non-native Arabic learners, such as *Phonetic errors*, *Tanween errors,* and *Shadda errors.*

CHAPTER 4

ARABIC SPELL CHECKING, CORRECTION AND LANGUAGE MODEL

This chapter presents the main components of our prototype for Arabic spell checking and correction including the language model. This chapter is organized as follows. Section 4.1 describes the main components of Arabic spell checking and correction prototype. Section 4.2 presents the Arabic language model.

4.1. ARABIC SPELL CHECKING AND CORRECTION PROTOTYPE

This section outlines the main components of our prototype for Arabic spell checking and correction. The prototype consists of the pre-processing module, spell checking module and spell correction module. Figure 4.1 shows the proposed Arabic spell checking and correction prototype. The techniques used in each module are also presented.

The pre-processing module is used to read the input text or document that contains Arabic text that is to be spell checked. This module extracts the words using punctuation marks and spaces. Diacritics, numbers, symbols and punctuation marks are removed from text before processing.

The spell checking module detects the errors in the input text. It validates each word of the input text using a dictionary look-up, a morphological analyzer, and n-grams language models.

The dictionary look-up is used to compare each word of the input text with words in the dictionary. If that word is in the dictionary, then it is assumed as a correct word. Otherwise, it is considered as a misspelled word.

The morphological analyzer is used to check whether a word is a correct word or a misspelled word using the morphology of Buckwalter's morphological analyzer. Buckwalter Arabic Morphological Analyzer (BAMA) has three components: the lexicon, the compatibility tables and the morphological analysis algorithm[3]. The lexicon has three lexicon files: dictPrefixes, (which contains all Arabic prefixes and their concatenations), dictStems, (which contains all Arabic stems), and dictSuffixes, which contains all Arabic suffixes and their concatenations. There are three compatibility tables to validate the prefix-stem, stem-suffix, and prefix-suffix combinations. Buckwalter suggests a simple rule based morphology analyzer for Arabic language. Buckwalter Arabic Morphological Analyzer (BAMA) is described in Chapter 5.

N-grams language models on character level are used to detect the spelling errors. Character n-grams are a subsequence of n characters of a word. Character bi-grams, tri-

[3] http://www.qamus.org/morphology.htm

grams and quad-grams analysis are used to detect spelling errors. Character n-grams work by checking each character n-gram in an input word with the table of n-gram statistics. If the words have low frequent n-grams, they are detected as probable misspelled words.

The spell correction module corrects the errors that are detected in the spell checking module. It has the following steps for each errors word: (1) generate a list of candidate words; (2) rank the candidate words; and (3) correct the error word.

After detecting the misspelled word, edit distance technique is used to generate the candidate words from a dictionary that are similar to the misspelled word. Language models (word n-grams) are used to rank the candidate words in a descending order according to their probabilities.

The spell correction module uses edit distance technique and language models (word n-grams). The details of the implementation of these modules are addressed in chapters 5 and 6.

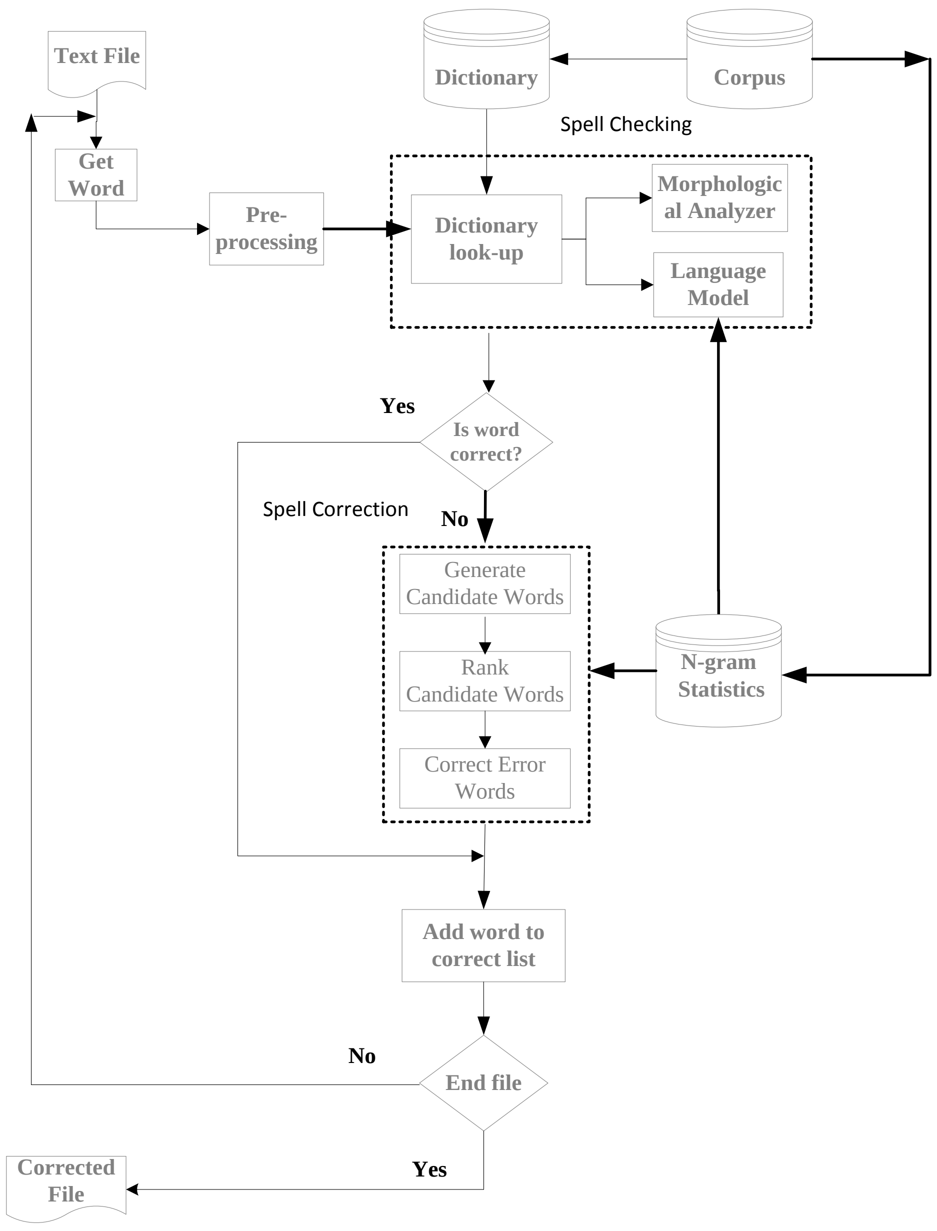


Figure 4.1 Arabic Spell Checking and Correction prototype

Figure 4.2 shows the implementation structure of Arabic spell checking and correction modules.

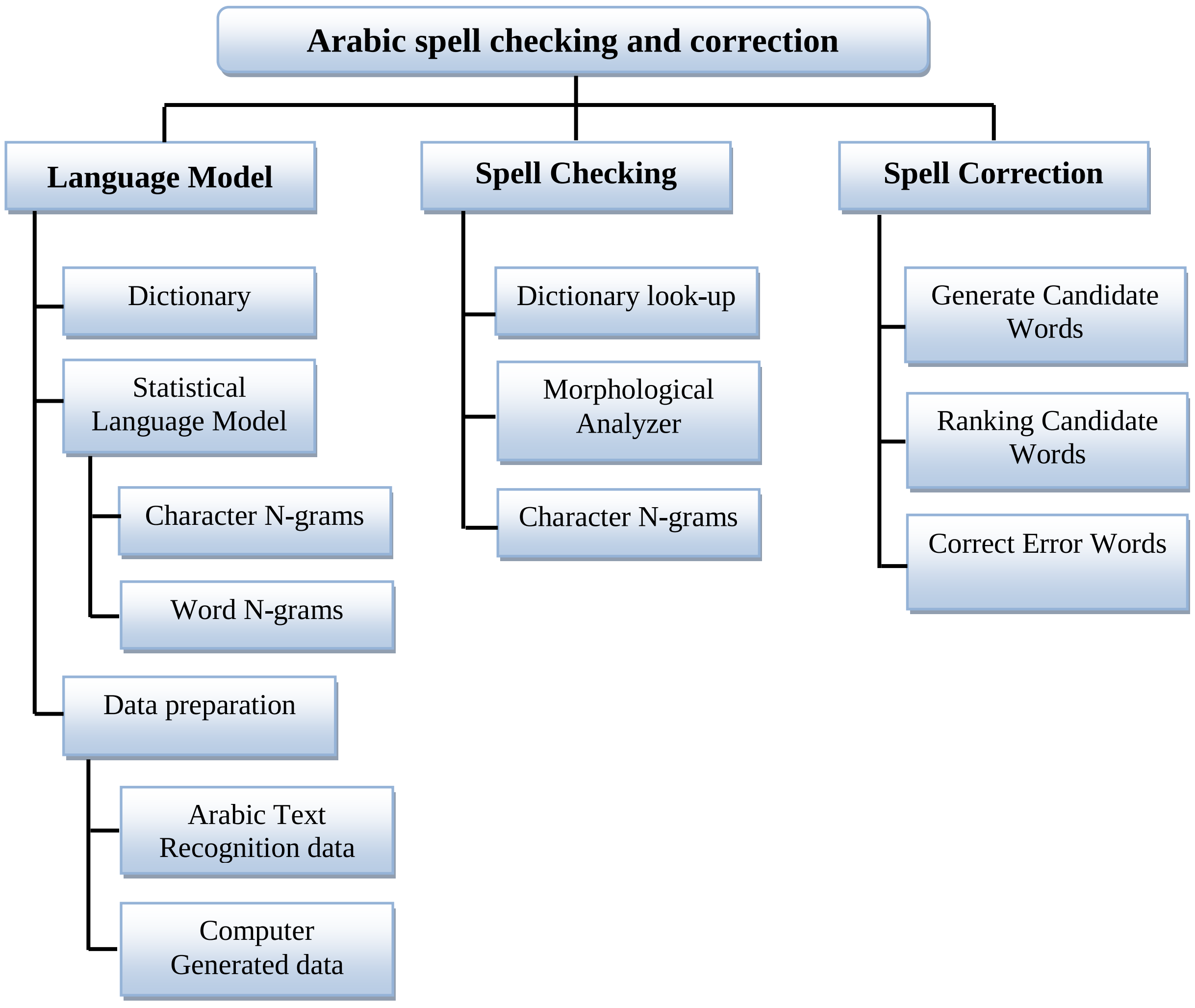


Figure 4.2 Implementation structure of Arabic Spell Checking and Correction

4.2. ARABIC LANGUAGE MODEL

In the following sections, we will present the components of the Arabic language model.

4.2.1. Corpus

In order to do spell checking and correction, we need a large data set (corpus). Therefore, we collected Arabic texts from different subjects such as news, short stories, and books. In addition, we used Arabic Gigaword Third Edition, Corpus of Contemporary Arabic (CCA) and Watan-2004 corpus.

- **Arabic Gigaword Third Edition**

Arabic Gigaword is a big and rich corpora compiled from different sources of Arabic newswire that includes six distinct sources: Agency France Press, Al Hayat News Agency, Al Nahar News Agency, Xinhua News Agency, Ummah Press and Assabah News Agency. In our corpus, we selected Al Hayat News Agency for the year 2005. It consists of 12 files distributed in 12 subfolders. The total size of the corpus exceeds 25 MB.

- **Corpus of Contemporary Arabic (CCA)**

The Corpus of Contemporary Arabic, abbreviated as CCA is the outcome of the MSc thesis of Latifa Al-Sulaiti (Al-Sulaiti 2004). She mainly derived texts from websites. A small amount of spoken texts is present, too. The corpus is reported to cover subjects of: autobiography, short stories, children's stories, economics, education, health and medicine, interviews, politics, recipes, religion, sociology, science (parts A and, newly, B), sports, tourist and travel, and spoken (sports, entertainment, education). To be noted, we did not use all files from this corpus because some text files contain distorted text. Figure 4.3 shows an example of text distorted in some files.

هثاية أخرى أث نشاط آخر ثما أخذ ثقتي ثله. لعل ذلث انعثس ثلث

Figure 4.3 An example of text distorted

We excluded three files from autobiography subject, six files from short stories subject, one file from children's stories subject, three files from interviews subject, and thirteen files from science subject. Table 4.1 shows the files that we excluded from the Corpus of Contemporary Arabic (CCA). Table 4.2 shows the part of CCA that we used.

Table 4.1 Excluded files from CCA

Subject	Files	Total
Autobiography	AUT01 - AUT15 - AUT18	3
Short stories	from S25 to S30	6
Children's stories	CHD10	1
Interviews	Int07- Int08 - Int09	3
Science	• from Sc11 to Sc19 • from Sc34 to Sc36 • Sc39	13

- **Watan-2004 corpus**

In addition, we used Watan-2004 corpus that contains about 20000 articles under one or more of the six following topics: Culture, Religion, Economy, Local News, International News and sports. To be noted, we did not use all topics. We ignored Religion topic because we found many repeated text files. The total size of the corpus exceeds 50 MB.

To sum up, the size of our corpus is 88.18 MB that contains around 10,808,714 words. Out of these words, 311,544 are distinct words. Table 4.2 shows the details of our corpus that are derived from each subject, and the number of words and distinct words in each subject.

Table 4.2 Corpus characteristic

	Reference	Subject	Size	Number of Words		Distinct Words
1	The Holy Quran	Quran	474 KB	77801		14950
2	Al Hayat 2005 Arabic Gigaword http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2007T40	News	25 MB	2294042		153327
3	(CCA) Corpus Latifa Al-Sulaiti http://www.comp.leeds.ac.uk/eric/latifa/research.htm	Varity of topics: Education , Science, Sports etc.	4.8 MB	750131		98680
4	**Watan-2004 corpus** http://sites.google.com/site/mouradabbas9/corpora	News	44.7 MB	Culture	1403904	138519
				International	862353	72709
				Economy	1469048	77945
				Local	1559248	88239
				Sports	1436808	76153
				Total	**6731361**	**453565**
5	http://www.saaid.net/book/list.php?cat=93 http://www.saaid.net/Warathah/arefe/14.htm	Stories	5.06 MB	106185		25876
6	http://www.almeshkat.net/books/list.php?cat=42 http://www.qassimy.com/book/view-513.html	Medicine	5.40 MB	612824		57897
7	http://www.almeshkat.net/books/open.php?cat=13&book=1104 http://books.bdr130.net/book/4 http://www.falestiny.com/news/1297	History	2.76 MB	236370		43130
		Total	**88.18 MB**	**10,808,714**		**311,544**

This data is used to build our dictionary and statistical language models (uni-gram, bi-gram and tri-gram). It is important to note that, after collecting Arabic texts, three tasks have been performed prior to build a dictionary and a language model:

- **Filtering the data**

  In all HTML files, we filtered Arabic text from those files by striping the HTML tags and extracting the raw text in the page.

- **Removing non Arabic characters**

  We considered removing all non Arabic characters like Latin alphanumeric characters and punctuation marks. Also, we removed all diacritic marks. Figure 4.4 shows an example of stripping the HTML tags and removing non Arabic characters.

- **Validating the data**

  We validated all the data in order to check if there are text files that contain misspelled words or have noise. Some of those errors occurred due to a missing space after a non-connectable character. Figure 4.5 shows samples of misspelled words in the corpus resulting from typos.

DOC id="HYT_ARB_20051010.0021" > < "type="story <HEADLINE>	
رغبة التعويض ستشعل المواجهة منذ البداية: النصر «الحزين» يستضيف الأهلي «الخاسر» في لقاء المصالحة	رغبة التعويض ستشعل المواجهة منذ البداية النصر الحزين يستضيف الأهلي الخاسر في لقاء المصالحة
<HEADLINE/> <DATELINE>	
الرياض - صالح الصالح الحياة - //05/10/11	الرياض صالح الصالح الحياة
<DATELINE/> <TEXT> <P>	
في لقاء يعني للضيف والمضيف الشيء الكثير، يستضيف النصر نظيره الأهلي في لقاء تم تأجيله من الجولة الثانية من مسابقة كأس دوري خادم الحرمين الشريفين.	في لقاء يعني للضيف والمضيف الشيء الكثير يستضيف النصر نظيره الأهلي في لقاء تم تأجيله من الجولة الثانية من مسابقة كأس دوري خادم الحرمين الشريفين
<P/> <P>	
النصر يدرك جيداً أن خسارته هذا اللقاء ستزيد من خطورة وضعه في الدوري منذ البداية، خصوصاً أنه خسر مواجهته الوحيدة أمام القادسية في الدمام بهدفين لهدف، ويعاني من ضغوطات إدارية وجماهيرية لا حدود لها.	النصر يدرك جيدا أن خسارته هذا اللقاء ستزيد من خطورة وضعه في الدوري منذ البداية خصوصا أنه خسر مواجهته الوحيدة أمام القادسية في الدمام بهدفين لهدف ويعاني من ضغوطات إدارية وجماهيرية لا حدود لها
<P/>	

Figure 4.4 A sample of an article from the Al-Hayat corpus

صلاحالحياة	الطاحنبعد	وباشحون	الةوراء	التقىيم	الصحافةوكذلك

Figure 4.5 Examples of misspelled words in the corpus resulting from typos

4.2.2. Dictionary

Our dictionary is derived from a large corpus of tokenized words**.** So, we extracted all Arabic distinct words from our corpus to build the dictionary. As a result, our dictionary consists of more than 311,000 distinct words without punctuation marks and diacritics. Figure 4.6 shows the steps we followed to convert the corpus to a dictionary.

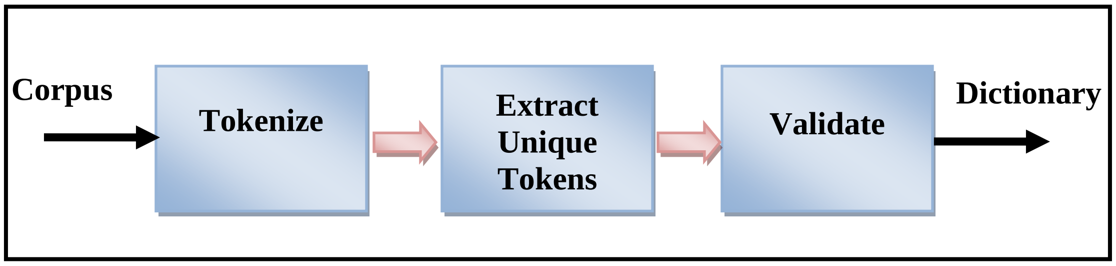


Figure 4.6 General scheme to convert corpus to dictionary

The first block, tokenize, is responsible for extracting all words from the corpus. While, the second block is responsible for keeping only one unique instance of every token. The third block is responsible for checking every unique token manually if it is a misspelled word and correcting them and finally, the outputs are aggregated to produce the word dictionary.

4.2.3. Statistical Language Models

A statistical language model is a probability distribution that estimates probabilities for word sequences $P(w_1 \ldots w_i)$, over the documents in the corpus. Language models are useful in many Natural Language Processing (NLP) applications (Jurafsky & Martin 2009), such as speech recognition (Oparin 2008), machine translation, (Raab 2006) (Brants et al. 2007), information retrieval (Zhai 2008) and spelling correction (Zhuang et al. 2004) (Islam & Inkpen 2009a) (Dalkilic & Cebi 2009) (Ahmed et al. 2009). The most widely used statistical language models are n-gram language models (Chen & Goodman 1998).

N-grams language models are either n-character subsequences of characters or of words, where n can be equal to one, two, three or more. One-character n-grams are referred to as uni-grams; two-character n-grams are referred to as bi-grams; and three-character n-grams are referred to as tri-grams (Kukich 1992).

There are many LM toolkits that are used to build and evaluate statistical language models. The most known are CMU Statistical Language Modeling (SLM) Toolkit[4], SRI Language Modeling (SRILM) Toolkit[5] and IRST Language Modeling Toolkit (IRSTLM)[6]. These LMs estimate n-gram probabilities from a text corpus and compute the probability of an n-gram. All these LMs run on Unix/Linux platforms except SRILM

[4] http://www.speech.cs.cmu.edu/SLM_info.html
[5] http://www.speech.sri.com/projects/srilm
[6] http://sourceforge.net/projects/irstlm/files/irstlm/

which runs on both Unix/Linux, and Windows platforms using Cygwin. We used IRST Language Modeling Toolkit (IRSTLM) (Federico et al. 2008) to build word n-gram language models. IRST Language Modeling Toolkit is used to extract the dictionary and the n-gram statistics from a corpus. We used IRST Language Modeling Toolkit Version 5.20.00 because it is a new version and it's feature algorithms and data structures are suitable to estimate, store, and access very large LMs (Federico et al. 2008). Furthermore, it requires less memory than SRILM (Federico & Cettolo 2007).

4.2.3.1. Character N-grams

In order to build character n-grams, a dictionary or a large corpus of text is required. Therefore, we used our dictionary to obtain n-gram table frequencies. Our statistics are classified into frequencies and sorted to character bi-grams, tri-grams and quad-grams. An example of bi-gram, tri-gram and quad-gram analysis of the word "*قاموس*" is as follows:

Bi-gram:	* ق	قا	ام	مو	وس	س *
Tri-gram:	* قا	قام	امو	موس	وس *	
Quad-gram:	* قام	قامو	اموس	موس *		

Space, marked with *, is also included in bi-gram, tri-gram and Quad-gram.

Table 4.3 shows the total and distinct number for bi-grams, tri-grams and quad-grams that we extracted from our dictionary.

Table 4.3 Character N-grams

Number of	Number	Distinct
Bi-grams	2,634,535	1,110
Tri-grams	2,276,731	18,633
Quad-gram	1,918,927	129,053

4.2.3.2. Word N-grams

Word n-grams are a sequence of *r* consecutive words in text. The probability of a sequence *s* of *r* words $w_1, w_2,..w_r$ is given by (Chen & Goodman 1998):

$$p(s) = p(w_1)\, P(w_2 \mid w_1)\, p(w_3 \mid w_1 w_2) \ldots p(w_r \mid w_1 \ldots w_{r-1})$$
$$= \prod_{i=1}^{r} p(w_i \mid w_1 \ldots w_{i-1})$$

Using the bi-gram models to compute higher n-grams, the probability is given by (Chen & Goodman 1998):

$$p(s) = \prod_{i=1}^{r} p(w_i \mid w_1 \ldots w_{i-1}) \approx \prod_{i=1}^{r} p(w_i \mid w_{i-1})$$

In order to build word n-grams, a large corpus of text is required. Therefore, we used our corpus to obtain uni-grams, bi-grams and tri-grams probabilities. Figure 4.7 shows an example of word bi-grams. The bi-grams are listed, followed by the actual text of the bi-gram. All probabilities are in logarithm (base 10) format (Stolcke 2002). Table 4.4 shows the data sizes of the corpus.

The problem of higher grams is due to limited data. The 4-gram and 5-gram will be mostly with small probability or zero occurrences. Hence we restrict our work for applicable 3-gram.

Table 4.4 Data sizes of the corpus (Word N-grams)

Number of	Number	Size on disk (MB)
Uni-grams	311,544	11.5 MB
Bi-grams	4,836,965	198 MB
Tri-grams	8,585,266	358 MB

Probabilities in Logarithm (base 10)	Bi-grams
-3.476687	كتب الدراسة
-2.777717	كتب الرحالة
-3.476687	كتب مقالاته
-2.999565	كتب العقاد
-3.476687	كتب القصيدة
-3.476687	كتب المنطق
-1.713259	كتب سالم
-2.999565	كتب السيرة
-3.476687	كتب فلسفية
-3.175657	كتب المدارس
-3.476687	كتب باللهجة
-3.175657	كتب المؤرخين
-3.476687	كتب بالروسية

Figure 4.7 Sample language model generated by IRSTLM

4.2.4. Data Preparation

This section describes the data that is used to test our Arabic spell checking and correction prototype as shown in Figure 4.8.

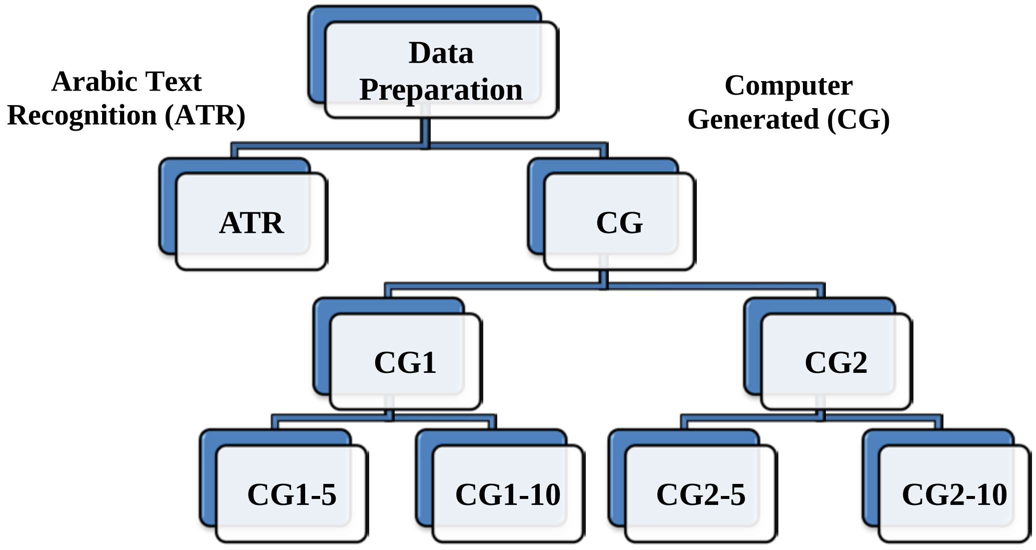


Figure 4.8 Data preparation

4.2.4.1. Arabic Text Recognition Data

In order to test our spell checking and correction prototype, two types of data sets are prepared. One set, Arabic Text Recognition (ATR) data, which is generated from an OCR system developed at KFUPM (Mahmoud & AlMuhtaseb 2010). The input to the system is part of scanned data from an old medical book titled Al-Jami by Ibn-Al-Bitar. The data consists of 3229 words that have 345 misspelled words.

4.2.4.2. Computer Generated Data

The second set, Computer Generated (CG) data with errors, which is prepared by taking a normal correct text and randomly introducing three types of errors (*insert, delete and replace*). The insert operation adds a random character at a random location of a randomly selected word; the delete operation deletes a randomly selected character from a randomly selected word; and replace operation replaces a randomly selected character from a randomly selected word by a random character. This data is further classified into two data types regarding the numbers of introduced character errors. The first one is called CG1, which has single-character errors per word. The second is called CG2, which has two-character errors per word. Each data set consists of 6665 words. This dataset was prepared to address the case where the OCR system recognition rate is too low. Hence, resulting in an output that has errors that are semi-random.

Injecting 5% and 10% errors into CG1 resulted in two data sets CG1-5 and CG1-10, respectively. Similarly, injecting 5% and 10% errors into CG2 resulted in another two data sets CG2-5 and CG2-10, respectively.

We applied an algorithm that selects a word randomly. Then it selects a character of the word randomly and randomly applies the *insert, delete and replace* errors. Figure 4.9 shows the algorithm for introducing the three types of errors *(insert, replace and delete).* After injecting 5% and 10% errors, we manually labeled the spelling mistakes in order to know the total number of true spelling errors. So, we checked all words against well known Arabic dictionaries such as Mukhtar Al Sehah and Lessan Al-Arab. In addition, some words were validated by an expert in Arabic language. As a result, out of 6665 words in the data set, there are 282 misspelled words. This data set has 110 inserted characters, 120 replaced characters and 52 deleted characters in CG1-5; 545 misspelled words, 221 inserted characters, 220 replaced characters and 104 deleted characters in CG1-10. In addition, 271 misspelled words in CG2-5 with 144 inserted, 88 replaced and 39 deleted characters; and 502 misspelled words, 221 inserted, 207 replaced and 74 deleted characters in CG2-10. Table 4.5 summarizes the numbers of introduced character errors**.**

Table 4.5 Introducing three types of errors

CG Data	**Error Type**			**Total**
	Inserted	**Replaced**	**Deleted**	
CG1-5	110	120	52	**282**
CG1-10	221	220	104	**545**
CG2-5	144	88	39	**271**
CG2-10	221	207	74	**502**

```
Input: normal correct text
Output: Text with errors
Begin
  1. Select a word randomly from the input text;
  2. Select a character randomly from the selected word;
  3. Select a character randomly from Arabic alphabet;
  4. Select an operation randomly to generate different types of errors
       (insert, replace and delete)
     If operation is insert then
            Insert_character        */  insert a character that we selected on step 3
                                            at character location on step 2   */
     elseif operation is replace then
            Replace_character       */  replace a character that we selected on step
                                            3 by a character on step 2   */
     else
            Delete_character        */  delete a character that we selected on step 2 */
     End if
End
```

Figure 4.9 Algorithm for introducing one of three different types of errors

CHAPTER 5

ARABIC SPELL CHECKING PROTOTYPE

In the previous chapter, we have presented our prototype for Arabic spell checking and correction and the language model. In this chapter, we describe the Arabic spell checking prototype. Then, we give details of the evaluation measures used to determine the Arabic spell checking performance.

Arabic spell checking detects the errors in the input text. This prototype uses the following methods:

- Dictionary look-up;
- Morphological analyzer;
- Language model (character N-grams)

5.1. DICTIONARY LOOK-UP

Dictionary look-up is the main component of the spell checking which will take a word and check whether it is a correct word or a misspelled word. It is used to compare each word of the input text with words in the dictionary. If that word is in the dictionary, then it is assumed as a correct word. Otherwise, it is considered as a misspelled word.

5.2. MORPHOLOGICAL ANALYZER

Morphological analyzer is used to check whether a word is a correct or a misspelled word using the morphology of Buckwalter's morphological analyzer (Buckwalter 2002). Buckwalter Arabic Morphological Analyzer (BAMA) has three components: the lexicon, the compatibility tables and the morphological analysis algorithm. The lexicon has three lexicon files: dictPrefixes, (which contains all Arabic prefixes and their concatenations), dictStems, (which contains all Arabic stems), and dictSuffixes which contains all Arabic suffixes and their concatenations. There are three compatibility tables to validate the prefix-stem, stem-suffix, and prefix-suffix combinations.

Buckwalter suggests a simple rule based morphology analysis for the Arabic language. The morphology analysis algorithm uses three parts of the input word: the prefix, the stem, and the suffix. Initially the whole word is considered as a stem and both the prefix and suffix parts are empty. Then the stem is checked against the dictionary of stems, and if it exists then the stem is returned with the correct tag acquired from the stems dictionary. If the stem is not found then, the last letter is removed from the input word and is added to the suffix part and the new stem replaces the existing stem in the stem part. The suffix is then looked up in the suffixes dictionary and in addition to looking up the new stem in the stems dictionary. If both stem and suffix are found, the rules file is used to find whether it is morphologically correct to add the suffix to the stem using their tags. For instance, the word ”كتبه” has the stem "كتب" which has the tag

“PV” and the suffix is “ه” with the tag PVSuff-ah and there exists a rule in the stem-suffix rules file that combines PV with PVSuff-ah which is simply stated as PV→PVSuff-ah. The prefixes are checked in a similar way and the process of checking for stems, prefixes, and suffixes in the input word continues until the number of letters in the suffixes exceeds 6 or the number of letters in the prefixes exceeds 4 or all possibilities are checked. Those conditions are based on the facts that the Arabic suffixes cannot exceed 6 letters and similarly the prefixes cannot exceed 4 letters. The following algorithms simulate the process of the Buckwalter Arabic morphological analyzer algorithm to make it easier to comprehend.

Algorithm 5.1 Buckwalter Arabic Morphological Analyzer

```
Input:    word
Output:   valid word or not
Begin
   Accepted←CheckStem (word)                /*look up the word in Stem Dictionary*/
   if not (Accepted) then
      Accepted← CheckStemSuff (word)        /* segment word into SUFF & STEM and
                                               check Buckwalter compatibility rules*/
        if not (Accepted) then
           Accepted← CheckPrefStem (word)   /* segment word into PREF & STEM and
                                               check Buckwalter compatibility rules
             if not (Accepted) then
                Accepted← CheckPrefStemSuff (word)   /* segment word into PREF, STEM
                                                        & SUFF and check Buckwalter
                                                        compatibility rules*/
             end if
        end if
   end if
End
```

Algorithm 5.2 CheckStemSuff for input word

```
Input : word
Output: Accepted Word or not
Begin
  [STEM  SUFF]← GetallStemSuff(word)     /*  segment word into SUFF and  STEM */
  look up the SUFF in SuffixDictionary
  look up the STEM in StemDic
  if both STEM and SUFF are found then
      Accepted←check Buckwalter compatibility rules
  end if
End
```

Algorithm 5.3 CheckPrefStem for input word

```
Input : word
Output: Accepted Word or not
Begin
  [PREF STEM]← GetallPrefStem (word)      /* segment word into PREF and STEM */
  look up the PREF in PrefDic
  look up the STEM in StemDic
  if both PREF and STEM are found then
      Accepted←check Buckwalter compatibility rules
  end if
End
```

Algorithm 5.4 CheckPrefStemSuff for input word

```
Input : word
Output: Accepted Word or not
Begin
  [PREF STEM  SUFF]← GetallPrefStemSuff(word)     /* segment word into PREF,
                                                      SUFF and STEM */
  look up the PREF in PrefDic
  look up the SUFF in SuffDic
  look up the STEM in StemDic
  if STEM , PREF and SUFF are found then
     Accepted←check Buckwalter compatibility rules
  end if
End
```

To illustrate the process of the Buckwalter Arabic Morphological Analyzer (BAMA) algorithm, there are three cases:

Case 1: The word has only the stem preceded by the prefix.

The word '**الكتاب**' has compatibility rules for the combination of the stem and the prefix rules as shown in Figure 5.1. Therefore, this word is accepted as a correct word.

الكتاب

Suffix _Rule	Stem_Rule	Suffix	Stem
No	No	ب	الكتا
No	No	اب	الكت
No	No	تاب	الك
No	Yes	كتاب	ال
No	Yes	لكتاب	ا

Stem_Rule	Prefix_Rule	Stem	Prefix
No	Yes	لكتاب	ا
Yes	Yes	كتاب	ال

Figure 5.1 Illustration of the BAMA algorithm when the word has both the prefix and the stem

Case 2: The word has only the stem followed by the suffix.

The word **'كتابه'** has compatibility rules for the combination of the stem and the suffix rules as shown in Figure 5.2. Therefore, this word is accepted as a correct word.

كتابه

Suffix_Rule	Stem_Rule	Suffix	Stem
Yes	Yes	ه	كتاب

Figure 5.2 Illustration of the BAMA algorithm when the word has both the stem and the suffix

Case 3: The word has the prefix, the stem and the suffix.

The word **'المؤمنون'** has compatibility rules for the combination of the prefix, the stem and the suffix rules as shown in Figure 5.3. Therefore, this word is accepted as a correct word.

المؤمنون

Suffix_Rule	Stem_Rule	Prefix_Rule	Suffix	Stem	Prefix
Yes	No		ن	المؤمنو	
Yes	No		ون	المؤمن	
No	No		نون	المؤم	
No	No		منون	المؤ	
No	Yes		ؤمنون	الم	
No	Yes		مؤمنون	ال	

Suffix_Rule	Stem_Rule	Prefix_Rule	Suffix	Stem	Prefix
	No	Yes		لمؤمنون	ا
	No	Yes		مؤمنون	ال
	No	No		ؤمنون	الم
	No	No		منون	المؤ
	Yes	No		نون	المؤم

المؤمنو

Suffix_Rule	Stem_Rule	Prefix_Rule	Suffix	Stem	Prefix
Yes	No	Yes	ن	لمؤمنو	ا
Yes	No	Yes	ن	مؤمنو	ال
Yes	No	No	ن	ؤمنو	الم
Yes	Yes	No	ن	منو	المؤ

المؤمن

Suffix_Rule	Stem_Rule	Prefix_Rule	Suffix	Stem	Prefix
Yes	No	Yes	ون	لمؤمن	ا
Yes	Yes	Yes	ون	مؤمن	ال

Figure 5.3 Illustration of the BAMA algorithm when the word has the prefix, the stem and the suffix

5.3. CHARACTER N-GRAMS

Character n-grams are used to detect spelling errors. Character n-grams work as follows: for each character n-gram in an input word, a pre-compiled table of n-gram statistics is searched to determine its existence and its frequency. If words have low frequent n-grams, they are detected as probable misspelled words (Morris & Cherry 1975), (E. Zamora et al. 1981).

A table of character n-gram statistics is pre-compiled from our dictionary. These statistics are in the form of probability of character n-grams occurrence. The simplest form of character n-gram is a bi-gram array; it is a two dimensional array of size 36x36 whose elements represent all possible two-letter combinations of the Arabic alphabet. An example of bi-gram array is shown in Table 5.1. It reads Y after X, for example,"صث" is not a valid bi-gram because it has a probability of 0.000, (i.e. it never occurred in the text). However, the bi-gram" صب" is a valid bi-gram because it has a probability of 0.00054.

[X,Y] = Read Y after X

		Y										
		ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث
	ء	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00030	0.00000	0.00010	0.00007	0.00000
	آ	0.00001	0.00000	0.00000	0.00001	0.00000	0.00003	0.00001	0.00006	0.00001	0.00009	0.00005
	أ	0.00000	0.00000	0.00002	0.00002	0.00000	0.00002	0.00001	0.00076	0.00008	0.00074	0.00026
	ؤ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00012	0.00001	0.00001	0.00006	0.00004
	إ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00025	0.00000	0.00010	0.00006
	ئ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00017	0.00015	0.00021	0.00018	0.00001
	ا	0.00280	0.00001	0.00001	0.00038	0.00001	0.00336	0.00001	0.00445	0.00041	0.01323	0.00069
	ب	0.00001	0.00005	0.00077	0.00005	0.00041	0.00010	0.01154	0.00045	0.00108	0.00262	0.00023
	ة	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	ت	0.00000	0.00006	0.00089	0.00014	0.00000	0.00008	0.00343	0.00206	0.00018	0.00157	0.00031
	ث	0.00000	0.00000	0.00002	0.00000	0.00000	0.00000	0.00074	0.00018	0.00014	0.00012	0.00001
	ج	0.00000	0.00001	0.00010	0.00000	0.00000	0.00008	0.00298	0.00062	0.00035	0.00059	0.00004
	ح	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00287	0.00075	0.00053	0.00150	0.00017
	خ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00127	0.00050	0.00010	0.00094	0.00002
	د	0.00002	0.00001	0.00012	0.00003	0.00000	0.00008	0.00478	0.00038	0.00098	0.00101	0.00027
	ذ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00069	0.00034	0.00008	0.00009	0.00000
	ر	0.00004	0.00005	0.00034	0.00015	0.00000	0.00021	0.00985	0.00226	0.00196	0.00279	0.00025
X	ز	0.00002	0.00000	0.00002	0.00001	0.00000	0.00004	0.00189	0.00022	0.00024	0.00020	0.00000
	س	0.00000	0.00001	0.00040	0.00012	0.00000	0.00011	0.00388	0.00123	0.00038	0.00730	0.00000
	ش	0.00000	0.00002	0.00009	0.00003	0.00000	0.00007	0.00229	0.00044	0.00020	0.00077	0.00000
	ص	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00209	0.00054	0.00019	0.00015	0.00000
	ض	0.00000	0.00001	0.00001	0.00000	0.00000	0.00001	0.00135	0.00020	0.00021	0.00017	0.00000
	ط	0.00001	0.00001	0.00005	0.00002	0.00000	0.00010	0.00220	0.00075	0.00032	0.00021	0.00000
	ظ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00039	0.00003	0.00008	0.00007	0.00000
	ع	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00408	0.00107	0.00087	0.00185	0.00022
	غ	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00129	0.00021	0.00009	0.00032	0.00003
	ف	0.00001	0.00003	0.00081	0.00002	0.00006	0.00011	0.00571	0.00025	0.00070	0.00232	0.00004
	ق	0.00000	0.00000	0.00001	0.00000	0.00000	0.00000	0.00389	0.00096	0.00074	0.00152	0.00001
	ك	0.00000	0.00002	0.00019	0.00001	0.00004	0.00002	0.00409	0.00070	0.00027	0.00131	0.00026
	ل	0.00000	0.00026	0.00335	0.00004	0.00176	0.00005	0.01247	0.00364	0.00150	0.00737	0.00043
	م	0.00000	0.00007	0.00020	0.00044	0.00000	0.00010	0.00787	0.00113	0.00115	0.00427	0.00047
	ن	0.00000	0.00001	0.00006	0.00003	0.00000	0.00006	0.00966	0.00115	0.00104	0.00330	0.00013
	ه	0.00000	0.00000	0.00001	0.00001	0.00000	0.00001	0.01474	0.00042	0.00016	0.00035	0.00001
	و	0.00011	0.00013	0.00258	0.00002	0.00069	0.00007	0.02252	0.00395	0.00029	0.00528	0.00049
	ي	0.00014	0.00001	0.00022	0.00020	0.00000	0.00027	0.00746	0.00230	0.00616	0.00534	0.00040
	ى	0.00011	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 5.1 An example of the character frequency bi-gram array

5.4. EXPERIMENTAL RESULTS

5.4.1. Evaluation Measures

Our evaluation methodology is based on the common Natural Language Processing (NLP) measures i.e. recall, precision and F_1-measure. These measures are classically used in information retrieval. They are employed in evaluating our spell checking.

- **Recall**

The recall rate is the fraction of the errors in the text that are correctly detected. Namely, the total number of the actual misspelled words that are detected by the prototype divided by the total number of misspelled words. Equation 5.1 shows the error detection recall rate (the actual detected misspelled words compared with all misspelled words) (Liang 2008).

$$\boldsymbol{Recall = \frac{Number\ of\ actual\ misspelled\ words\ detected}{Number\ of\ all\ misspelled\ words\ in\ the\ data}\ x\ 100} \qquad (\mathbf{5.1})$$

- **Precision**

The precision rate is the percentage of the total number of the actual misspelled words that are detected by the prototype to the total number of detected words. Equation 5.2 shows the precision rate (Liang 2008).

$$\boldsymbol{Precision} = \frac{\boldsymbol{Number\ of\ actual\ misspelled\ words\ detected}}{\boldsymbol{Total\ number\ of\ detected\ words}}\ \boldsymbol{x\ 100} \qquad (\mathbf{5.2})$$

- **F$_1$-measure**

The F$_1$-measure is a harmonic measure that combines recall and precision into one single value. In general, the F$_1$-measure favors a balance between precision and recall. Equation 5.3 shows the F$_1$-measure (Sasaki 2007).

$$\mathbf{F_1 - measure} = \frac{\boldsymbol{2 * Precision * Recall}}{\boldsymbol{Precision + Recall}}\ \boldsymbol{x\ 100} \qquad (\mathbf{5.3})$$

5.4.2. Spell Checking Performance

In this section, we present the results for spell checking with respect to the aforementioned evaluation measures.

5.4.2.1. Spell Checking using Buckwalter's Arabic Morphological Analyzer (BAMA)

- **Arabic Text Recognition (ATR) data**

We used Buckwalter's Arabic morphological analyzer (BAMA) to detect the spelling errors of the ATR data (a total of 3229 words with 345 misspelled words). Using BAMA, we detected 466 words as misspelled words. Out of these words, 342 words are actual misspelled words (124 words are reported as misspelled words while they are correct words). This is due to the nature of the test data, which is taken from an old medical book. Moreover, BAMA dictionary stems does not contain all Arabic stems, such as "تتكرج", "القافسيا", "السيلقون" and "الأسفيداج". BAMA is unable to detect 3 misspelled words, "الأرضى ", " التى" , "العى" , of a total of 345 words. This is due to the normalizing Alef maqsoura to Ya { ى →ي }. BAMA achieved 99.13%, 73.39% and 84.34% recall, precision, and F_1-measure, respectively. Table 5.2 shows the spell checking results on the ATR data using BAMA. Table 5.38 shows the results compared to other methods.

Table 5.2 Spell checking results on ATR data using BAMA

Total misspelled words	345
Total detected	466
Actual misspelled detected	342
Actual misspelled undetected	3
Correct word detected as misspelled	124
Recall	**99.13**
Precision	**73.39**
F_1-measure	**84.34**

- **Computer Generated (CG) data**

We used Buckwalter's Arabic morphological analyzer to detect the spelling errors of our Computer Generated (CG) data (a total of 6665 words).

  - **CG1-5 data**

Using BAMA on CG1-5 data, we detected 316 words as misspelled words. Out of these words, 280 words are actual misspelled words (36 words are reported as misspelled words while they are correct words). This is because BAMA dictionary stems doesn't contain all Arabic stems , such as "الجرافيك" , "ابوظبي "," الإسفلت "," التأهيلية" and " الملتيميديا ". BAMA is unable to detect 2 misspelled words, "وليو", "وه", of a total of 282 words. BAMA achieved 99.29%, 88.61% and 93.65% recall, precision and F_1-measure, respectively. Table 5.3 shows the spell checking results on CG1-5 data using BAMA. Table 5.39 shows the results compared to other methods.

Table 5.3 Spell checking results on CG1-5 using BAMA

Total misspelled words	282
Total detected	316
Actual misspelled detected	280
Actual misspelled undetected	2
Correct word detected as misspelled	36
Recall	**99.29**
Precision	**88.61**
F_1-measure	**93.65**

- **CG1-10 data**

Using BAMA on CG1-10 data, we detected 576 words as misspelled words. Out of these words, 544 words are actual misspelled words (32 words are reported as misspelled words while they are correct words). This is because BAMA dictionary stems doesn't contain all Arabic stems, such as "الولهانة", "ابوظبي", "واللاجدوى" and "المتسلطة". BAMA is unable to detect 1 misspelled word," يلى ", of a total of 545 words. This is due to normalizing Alef maqsoura to Ya { ى →ي }. BAMA achieved 99.82%, 94.44% and 97.06% in terms of recall, precision and F_1-measure, respectively. Table 5.4 shows the spell checking results on CG1-10 data using BAMA, including recall, precision and F_1-measure. Table 5.40 shows the results compared to other methods.

Table 5.4 Spell checking results on CG1-10 using BAMA

Total misspelled words	545
Total detected	576
Actual misspelled detected	544
Actual misspelled undetected	1
Correct word detected as misspelled	32
Recall	**99.82**
Precision	**94.44**
F_1-measure	**97.06**

- **CG2-5 data**

Using BAMA on CG2-5 data, we detected 307 words as misspelled words. Out of these words, 270 words are actual misspelled words (36 words are reported as misspelled words while they are correct words). This is because BAMA dictionary stems doesn't contain all Arabic stems, such as "الجرافيك", "التجزيء", "المتسم" and "المتسمة". BAMA achieved 100%, 88.27% and 93.77% recall, precision and F_1-measure, respectively. Table 5.5 shows the spell checking results on CG2-5 data using BAMA, including recall, precision and F_1-measure. Table 5.41 shows the results compared to other methods.

Table 5.5 Spell checking results on CG2-5 using BAMA

Total misspelled words	271
Total detected	307
Actual misspelled detected	271
Actual misspelled undetected	0
Correct word detected as misspelled	36
Recall	**100**
Precision	**88.27**
F_1-measure	**93.77**

- **CG2-10 data**

Using BAMA, we detected 538 words as misspelled words of CG2-10 data. Out of these words, 500 words are actual misspelled words (38 words are reported as misspelled words while they are correct words). This is because BAMA dictionary stems doesn't contain all Arabic stems , such as "جدوى" , "المتداخلة"," المتسلطة" and " الملتيميديا ". BAMA is unable to detect 2 misspelled words, "اخفا" and "وبينى"of a total of 502 words. BAMA achieved 99.60%, 92.94% and 95.15% in terms of recall, precision and F_1-measure, respectively. Table 5.6 shows the spell checking results on CG2-10 data using BAMA, including recall, precision and F_1-measure. Table 5.42 shows the results compared to other methods.

Table 5.6 Spell checking results on CG2-10 using BAMA

Total misspelled words	502
Total detected	538
Actual misspelled detected	500
Actual misspelled undetected	2
Correct word detected as misspelled	38
Recall	**99.60**
Precision	**92.94**
F_1-measure	**96.15**

5.4.2.2. Spell Checking using Dictionary Look-Up

- **Arabic Text Recognition (ATR) data**

The dictionary look-up is used to detect the spelling errors of ATR data (a total of 3229 words). Dictionary look-up detected 581 words as misspelled words. Out of these words, 343 words are actual misspelled words (238 words are reported as misspelled words while they are correct words). This is due to the nature of the test data, which is taken from an old medical book. Furthermore, the dictionary does not contain those words. Dictionary look-up is unable to detect 2 misspelled words, "شاينا", "فانك", of a total of 345 words. This is because the dictionary contains words translated from English language. Dictionary look-up achieved 99.42%, 59.04% and 74.08% recall, precision and F_1-measure respectively. Table 5.7 shows the spell checking results on ATR data using dictionary look-up. Table 5.38 shows the results compared to other methods.

Table 5.7 Spell checking results on CG1-5 using Dictionary look-up

Total misspelled words	345
Total detected	581
Actual misspelled detected	343
Actual misspelled undetected	2
Correct word detected as misspelled	238
Recall	**99.42**
Precision	**59.04**
F_1-measure	**74.08**

When we update our dictionary by adding the book "Al-Jami by Ibn-Al-Bitar ". We got the result as shown in Table 5.8. We can see the spell checking results in term of precision and F_1-measureare better than the results in Table 5.7.

Table 5.8 Spell Checking results on ATR data using Dictionary look-up after update the Dictionary

Total misspelled words	345
Total detected	455
Actual misspelled detected	343
Actual misspelled undetected	2
Correct word detected as misspelled	112
Recall	**99.42**
Precision	**75.38**
F_1-measure	**85.75**

- **Computer Generated (CG) data**

The dictionary look-up is used to detect the spelling errors of Computer Generated output (a total of 6665 words).

- **CG1-5 data**

Dictionary look-up detected 320 words as misspelled words of CG1-5 data. Out of these words, 281 words are actual misspelled words (39 words are reported as misspelled words while they are correct words). This is because the dictionary does not contain all words, such as "كأبحاث" , "وبأدوارها "," لأسواقنا" and "مستكشفين". Dictionary look-up is unable to detect 1 misspelled word, "يه", of a total of 282 words. This is because the dictionary contains words translated from English language misspelled words. Dictionary look-up achieved 99.65%, 87.81% and 93.36% recall, precision and F_1-measure respectively. Table 5.9 shows the spell checking results on CG1-5 data using dictionary look-up. Table 5.39 shows the results compared to other methods.

Table 5.9 Spell checking results on CG1-5 using Dictionary look-up

Total misspelled words	282
Total detected	320
Actual misspelled detected	281
Actual misspelled undetected	1
Correct word detected as misspelled	39
Recall	**99.65**
Precision	**87.81**
F_1-measure	**93.36**

- **CG1-10 data**

Dictionary look-up detected 591 words as misspelled words of CG1-10 data. Out of these words, 544 words are actual misspelled words (47 words are reported as misspelled words while they are correct words). This is because the dictionary does not contain all words, such as "الطامعون", "الجامدون", "فنقتبس" and "وإغرائه". Dictionary look-up is unable to detect 1 misspelled word, "يها", of a total of 545 words. Dictionary look-up achieved 99.82%, 92.05% and 95.77% recall, precision and F_1-measure respectively. Table 5.10 shows the spell checking results on CG1-10 data using dictionary look-up. Table 5.40 shows the results compared to other methods.

Table 5.10 Spell Checking results on CG1-10 using Dictionary look-up

Total misspelled words	545
Total detected	591
Actual misspelled detected	544
Actual misspelled undetected	1
Correct word detected as misspelled	47
Recall	**99.82**
Precision	**92.05**
F_1-measure	**95.77**

- **CG2-5 data**

Dictionary look-up detected 314 words as misspelled words of CG2-5 data. Out of these words, 270 words are actual misspelled words (44 words are reported as misspelled words while they are correct words). This is because the dictionary does not contain all those words, such as "فركزت", "مصوغا" and "وكقيادة". Dictionary look-up is unable to detect 1 misspelled word, "اوي", of a total of 271 words. Dictionary look-up achieved 99.63%, 85.99% and 92.31% recall, precision and F_1-measure respectively.

Table 5.11 shows the spell checking results on CG2-5 data using dictionary look-up. Table 5.41 shows the results compared to other methods.

Table 5.11 Spell checking results on CG2-5 using Dictionary look-up

Total misspelled words	271
Total detected	314
Actual misspelled detected	270
Actual misspelled undetected	1
Correct word detected as misspelled	44
Recall	**99.63**
Precision	**85.99**
F_1-measure	**92.31**

- **CG2-10 data**

Dictionary look-up detected 586 words as misspelled words of CG2-10 data. Out of these words, 501 words are actual misspelled words (85 words are reported as misspelled words while they are correct words). This is because the dictionary does not contain all those words, such as "التنمي", "وسجدات", "ميمن" and "وإسهامات". Dictionary look-up is unable to detect 1 misspelled word, "آمد", of a total of 502 words. Dictionary look-up achieved 99.80%, 85.49% and 92.10% recall, precision and F_1-measure respectively. Table 5.12 shows the spell checking results on CG2-10 data using dictionary look-up. Table 5.42 shows the results compared to other methods.

Table 5.12 Spell checking results on CG2-10 using Dictionary look-up

Total misspelled words	502
Total detected	586
Actual misspelled detected	501
Actual misspelled undetected	1
Correct word detected as misspelled	85
Recall	**99.80**
Precision	**85.49**
F_1-measure	**92.10**

5.4.2.3. Spell Checking using a combination of Dictionary Look-Up and BAMA

We used both dictionary look-up and BAMA to detect the spelling errors of Arabic text recognition (ATR) (a total of 3229 words) and CG data (a total of 6665 words). First, we use dictionary look-up to check each word. Then, if the word is out of dictionary, we use BAMA to check the word.

- **Arabic Text Recognition (ATR) data**

Using this data set a combination of Dictionary Look-up and BAMA is applied. We detected 347 words as misspelled words of ATR data. Out of these words, 340 words are actual misspelled words (7 words, "ويقشن", "ماق", "مادر", "لأسفيداج", "لأسفيداج", "لأسفيداج", "فرة", "صفنة", are reported as misspelled word while they are correct words). The combination of Dictionary Look-up and BAMA are unable to detect 5 misspelled words, "الأرضى", "التى", "العى", "شاينا","فانك", of a total of 345 words. The combination of Dictionary Look-up and BAMA achieved 98.55%, 97.98% and 98.27% recall, precision, and F1-measure, respectively. Table 5.13 shows the spell checking results on ATR data using the combination of Dictionary Look-up and BAMA. Table 5.38 shows the results compared to other methods.

Table 5.13 Spell checking results on ATR data using a combination of Dictionary look-up and BAMA

Total misspelled words	345
Total detected	347
Actual misspelled detected	340
Actual misspelled undetected	5
Correct word detected as misspelled	7
Recall	**98.55**
Precision	**97.98**
F_1-measure	**98.27**

- **Computer Generated (CG1-5) data**

Using this data set the combination of Dictionary Look-up and BAMA detected 281 words as misspelled words of CG1-5 data. Out of these words, 279 words are actual misspelled words (2 words, "وإقصائها", "الطامعون", are reported as misspelled words while they are correct words). The combination of Dictionary Look-up and BAMA are unable to detect 3 misspelled words, "وليو", " يه", " وه", of a total of 282 words. The combination of Dictionary Look-up and BAMA achieved 98.94%, 99.29% and 99.11% recall, precision and F_1-measure respectively. Table 5.14 shows the spell checking results on CG1-5 data using the combination of Dictionary Look-up and BAMA. Table 5.39 shows the results compared to other methods.

Table 5.14 Spell checking results on CG1-5 using a combination of Dictionary Look-up and BAMA

Total misspelled words	282
Total detected	281
Actual misspelled detected	279
Actual misspelled undetected	3
Correct word detected as misspelled	2
Recall	**98.94**
Precision	**99.29**
F_1-measure	**99.11**

- **Computer Generated (CG1-10) data**

Using this data set the combination of Dictionary Look-up and BAMA detected 549 words as misspelled words of CG1-10 data. Out of these words, 543 words are actual misspelled words (6 words, " مشكة", " ديثا", "الطامعون", " وإقصائها", " ذه ", " ذه ", are reported as misspelled words while they are correct words). The combination of Dictionary Look-up and BAMA are unable to detect 2 misspelled words, "يلى ", "يها ", of a total of 545 words. The combination of Dictionary Look-up and BAMA achieved 99.63%, 98.91% and 99.27% recall, precision and F_1-measure respectively. Table 5.15 shows the spell checking results on CG1-10 data using the combination of Dictionary Look-up and BAMA. Table 5.40 shows the results compared to other methods.

Table 5.15 Spell checking results on CG1-10 using a combination of Dictionary Look-up and BAMA

Total misspelled words	545
Total detected	549
Actual misspelled detected	543
Actual misspelled undetected	2
Correct word detected as misspelled	6
Recall	**99.63**
Precision	**98.91**
F_1-measure	**99.27**

- **Computer Generated (CG2-5) data**

Using this data set the combination of Dictionary Look-up and BAMA detected 272 words as misspelled words of CG2-5 data. Out of these words, 270 words are actual misspelled words (2 words, "للربية", "الطامعون", are reported as misspelled words while they are correct words). The combination of Dictionary Look-up and BAMA are unable to detect 1 misspelled word, "اوي" of a total of 271 words. The combination of Dictionary Look-up and BAMA achieved 99.63%, 99.26% and 99.45% recall, precision and F_1-measure respectively. Table 5.16 shows the spell checking results on CG2-5 data using the combination of Dictionary Look-up and BAMA. Table 5.41 shows the results compared to other methods.

Table 5.16 Spell checking results on CG2-5 using a combination of Dictionary Look-up and BAMA

Total misspelled words	271
Total detected	272
Actual misspelled detected	270
Actual misspelled undetected	1
Correct word Detected as misspelled	2
Recall	**99.63**
Precision	**99.26**
F_1-measure	**99.45**

- **Computer Generated (CG2-10) data**

Using this data set the combination of Dictionary Look-up and BAMA detected 505 words as misspelled words of CG2-10 data. Out of these words, 499 words are actual misspelled words (6 words, " وإقصائها", "أثجم", "الطامعون", "يمنن", "ديث", "يسيفا", are reported as misspelled words while they are correct words). The combination of Dictionary Look-up and BAMA are unable to detect 3 misspelled words, "اخفا", "وبينى", "آمد", of a total of 502 words. The combination of Dictionary Look-up and BAMA achieved 99.40%, 98.81% and 99.11% recall, precision and F_1-measure respectively. Table 5.17 shows the spell checking results on CG2-10 data using the combination of Dictionary Look-up and BAMA. Table 5.42 shows the results compared to other methods.

Table 5.17 Spell checking results on CG2-10 using a combination of Dictionary Look-up and BAMA

Total misspelled words	502
Total detected	505
Actual misspelled detected	496
Actual misspelled undetected	3
Correct word detected as misspelled	6
Recall	**99.40**
Precision	**98.81**
F_1-measure	**99.11**

5.4.2.4. Spell Checking using Language Model (Character N-grams)

Character n-grams (*bi-grams, tri-grams and quad-gram)* are used to detect spelling errors. Character n-grams work as follows: for each character n-gram in an input word, a pre-compiled table of n-gram statistics is searched to determine its existence and its frequency. If words have low frequent n-grams, they are detected as probable misspelled words (Morris & Cherry 1975), (E. Zamora et al. 1981).

For example, we have chosen the character tri-gram for an input word. The procedure that we used in this work to detect the input word if it is a correct word or not is as follows: first, we extract all character tri-grams from the input word. After that, we look up all its character tri-grams in the tri-gram statistics table. Finally, if all its tri-grams appear in the pre-compiled table of tri-gram statistics, then we say this word is a correct word. Otherwise, the word is reported as a misspelled word.

- **Arabic Text Recognition (ATR) data**

  - ***Character Bi-gram***

Using this data character *bi-gram* detected 202 words as misspelled words of ATR data. Out of these words, 21 words are actual misspelled words (181 words are reported as misspelled words while they are correct words). Character *bi-gram* is unable to detect 324 misspelled words of a total of 345 words. This is due to the fact that most of the combination of any two characters in Arabic language is valid word. Character *bi-gram* achieved 6.09%, 10.40% and 7.68% recall, precision and F_1-measure respectively. Table 5.18 shows the spell checking results on ATR data using Character *bi-gram*. Table 5.38 shows the results compared to other methods.

Table 5.18 Spell checking results on ATR data using character Bi-gram

Total misspelled words	345
Total detected	202
Actual misspelled detected	21
Actual misspelled undetected	324
Correct word detected as misspelled	181
Recall	**6.09**
Precision	**10.40**
F_1-measure	**7.68**

  - ***Character Tri-gram***

Character *tri-gram* detected 159 words as misspelled words of ATR data. Out of these words, 113 words are actual misspelled words (46 words are reported as misspelled words while they are correct words). Character *tri-gram* is unable to detect 226 misspelled words of a total of 339 words. This is because the most combination of any three characters in Arabic language is valid word. Character *tri-gram* achieved 33.33%, 71.07% and 45.38% recall, precision and F_1-measure respectively. Table 5.19 shows the spell checking results on ATR data using Character *tri-gram*. Table 5.39 shows the results compared to other methods. Table 5.38 shows the results compared to other methods. It is important to note that, tri-grams skip words with less than three characters. Hence, the total misspelled words are 339 misspelled words of a total of 345 words in ATR data.

Table 5.19 Spell checking results on ATR data using character Tri-gram

Total misspelled words	339
Total detected	159
Actual misspelled detected	113
Actual misspelled undetected	226
Correct word detected as misspelled	46
Recall	**33.33**
Precision	**71.07**
F_1-measure	**45.38**

- ***Character Quad-gram***

Character *quad-gram* detected 266 words as misspelled words of ATR data. Out of these words, 215 words are actual misspelled words (51 words are reported as misspelled words while they are correct words). Character *quad-gram* is unable to detect 107 misspelled words of a total of 322 words. Character *quad-gram* achieved 66.77%, 80.83% and 73.13% recall, precision and F_1-measure respectively. Table 5.20 shows the spell checking results on ATR data using Character *quad-gram*. Table 5.38 shows the results compared to other methods. It is important to note that, quad-grams skip words with less than four characters. Hence, the total misspelled words are 322 misspelled words of a total of 345 words in ATR data.

Table 5.20 Spell checking results on ATR data using character Quad-gram

Total misspelled words	322
Total detected	266
Actual misspelled detected	215
Actual misspelled undetected	107
Correct word detected as misspelled	51
Recall	**66.77**
Precision	**80.83**
F_1-measure	**73.13**

- ***Combination of Character n-grams***

The combination of character *n-grams* detected 292 words as misspelled words of ATR data. Out of these words, 221 words are actual misspelled words (71 words are reported as misspelled words while they are correct words). The combination of character *n-grams* is unable to detect 124 misspelled words of a total of 345 words. The combination of character *n-grams* achieved 64.06%, 75.68% and 69.39% recall, precision and F_1-measure respectively. Table 5.21 shows the spell checking results on ATR data using the combination of character *n-grams*.

Table 5.21 Spell checking results on ATR data using a combination of character n-grams

Total misspelled words	345
Total detected	292
Actual misspelled detected	221
Actual misspelled undetected	124
Correct word detected as misspelled	71
Recall	**64.06**
Precision	**75.68**
F_1-measure	**69.39**

- **Computer Generated (CG1-5) data**
  - ***Character Bi-gram***

Character *bi-gram* detected 345 words as misspelled words of CG1-5 data. Out of these words, 94 words are actual misspelled words (251 words are reported as misspelled words while they are correct words). Character *bi-gram* is unable to detect 188 misspelled words of a total of 282 words. This is due to the fact that most of the combination of any two characters in Arabic language is valid word. Character *bi-gram* achieved 33.33%, 27.25% and 29.98% recall, precision and F_1-measure respectively. Table 5.22 shows the spell checking results on CG1-5 data using Character *bi-gram*. Table 5.39 shows the results compared to other methods.

Table 5.22 Spell checking results on CG1-5 data using character Bi-gram

Total misspelled words	282
Total detected	345
Actual misspelled detected	94
Actual misspelled undetected	188
Correct word detected as misspelled	251
Recall	**33.33**
Precision	**27.25**
F_1-measure	**29.98**

- ***Character Tri-gram***

Character *tri-gram* detected 245 words as misspelled words of CG1-5 data. Out of these words, 136 words are actual misspelled words (109 words are reported as misspelled words while they are correct words). Character *tri-gram* is unable to detect 134 misspelled words of a total of 270 words. This is because the majority of combination of three characters in Arabic language is valid word. Character *tri-gram* achieved 50.37%, 55.51% and 52.82% recall, precision and F_1-measure respectively. Table 5.23 shows the spell checking results on CG1-5 data using Character *tri-gram*. Table 5.39 shows the results compared to other methods.

Table 5.23 Spell checking results on CG1-5 data using character Tri-gram

Total misspelled words	270
Total detected	245
Actual misspelled detected	136
Actual misspelled undetected	134
Correct word detected as misspelled	109
Recall	**50.37**
Precision	**55.51**
F_1-measure	**52.82**

- ***Character Quad-gram***

Character *quad-gram* detected 210 words as misspelled words of CG1-5 data. Out of these words, 185 words are actual misspelled words (25 words are reported as misspelled words while they are correct words). Character *quad-gram* is unable to detect 58 misspelled words of a total of 243 words. This is because a large amount of combination of any four characters in Arabic language may be a valid word. Character *quad-gram* achieved 76.13%, 88.10% and 81.68% recall, precision and F_1-measure respectively. Table 5.24 shows the spell checking results on CG1-5 data using Character *quad-gram*. Table 5.39 shows the results compared to other methods.

Table 5.24 Spell checking results on CG1-5 data using character Quad-gram

Total misspelled words	243
Total detected	210
Actual misspelled detected	185
Actual misspelled undetected	58
Correct word detected as misspelled	25
Recall	**76.13**
Precision	**88.10**
F_1-measure	**81.68**

- ***Combination of Character n-grams***

The combination of character *n-grams* detected 210 words as misspelled words of CG1-5 data. Out of these words, 185 words are actual misspelled words (25 words are reported as misspelled words while they are correct words). The combination of character *n-grams* is unable to detect 97 misspelled words of a total of 282 words. The combination of character *n-grams* achieved 65.60%, 88.10% and 75.20% recall, precision and F_1-measure respectively. Table 5.25 shows the spell checking results on CG1-5 data using the combination of character *n-grams*. Table 5.39 shows the results compared to other methods.

Table 5.25 Spell checking results on CG1-5 data using the combination of character n-grams

Total misspelled words	282
Total detected	210
Actual misspelled detected	185
Actual misspelled undetected	97
Correct word detected as misspelled	25
Recall	**65.60**
Precision	**88.10**
F_1-measure	**75.20**

- **Computer Generated (CG1-10) data**

  - ***Character Bi-gram***

Character *bi-gram* detected 423 words as misspelled words of CG1-10 data. Out of these words, 189 words are actual misspelled words (234 words are reported as misspelled words while they are correct words). Character *bi-gram* is unable to detect 356 misspelled words of a total of 545 words. This is due to the fact that most of the combination of any two characters in Arabic language is valid word. Character *bi-gram* achieved 34.68%, 44.68% and 39.05% recall, precision and F_1-measure respectively. Table 5.26 shows the spell checking results on CG1-10 data using Character *bi-gram*. Table 5.40 shows the results compared to other methods.

Table 5.26 Spell checking results on CG1-10 data using character Bi-gram

Total misspelled words	545
Total detected	423
Actual misspelled detected	189
Actual misspelled undetected	356
Correct word detected as misspelled	234
Recall	**34.68**
Precision	**44.68**
F_1-measure	**39.05**

- ***Character Tri-gram***

Character *tri-gram* detected 396 words as misspelled words of CG1-10 data. Out of these words, 285 words are actual misspelled words (111 words are reported as misspelled words while they are correct words). Character *tri-gram* is unable to detect 243 misspelled words of a total of 528 words. This is because the majority of combination of any three characters in Arabic language is valid word. Character *tri-gram* achieved 53.98%, 71.97% and 61.69% recall, precision and F_1-measure respectively. Table 5.27 shows the spell checking results on CG1-10 data using Character *tri-gram*. Table 5.40 shows the results compared to other methods.

Table 5.27 Spell checking results on CG1-10 data using character Tri-gram

Total misspelled words	528
Total detected	396
Actual misspelled detected	285
Actual misspelled undetected	243
Correct word detected as misspelled	111
Recall	**53.98**
Precision	**71.97**
F_1-measure	**61.69**

- ***Character Quad-gram***

Character *quad-gram* detected 402 words as misspelled words of CG1-10 data. Out of these words, 373 words are actual misspelled words (29 words are reported as misspelled words while they are correct words). Character *quad-gram* is unable to detect 117 misspelled words of a total of 490 words. This is because the large number of combination of any four characters in Arabic language may be a valid word. Character *quad-gram* achieved 76.12%, 92.79% and 83.63% recall, precision and F_1-measure respectively. Table 5.28 shows the spell checking results on CG1-10 data using Character *quad-gram*. Table 5.40 shows the results compared to other methods.

Table 5.28 Spell checking results on CG1-10 data using character Quad-gram

Total misspelled words	490
Total detected	402
Actual misspelled detected	373
Actual misspelled undetected	117
Correct word detected as misspelled	29
Recall	**76.12**
Precision	**92.79**
F_1-measure	**83.63**

- ***Combination of Character n-grams***

The combination of character *n-grams* detected 510 words as misspelled words of CG1-10 data. Out of these words, 411 words are actual misspelled words (99 words are reported as misspelled words while they are correct words). The combination of character *n-grams* is unable to detect 134 misspelled words of a total of 545 words. The combination of character *n-grams* achieved 75.41%, 80.59% and 77.91% recall, precision and F$_1$-measure respectively. Table 5.29 shows the spell checking results on CG1-10 data using the combination of character *n-grams*. Table 5.40 shows the results compared to other methods.

Table 5.29 Spell checking results on CG1-10 data using the combination of character n-grams

Total misspelled words	545
Total detected	510
Actual misspelled detected	411
Actual misspelled undetected	134
Correct word detected as misspelled	99
Recall	**75.41**
Precision	**80.59**
F$_1$-measure	**77.91**

- **Computer Generated (CG2-5) data**

- ***Character Bi-gram***

Character *bi-gram* detected 387 words as misspelled words of CG2-5 data. Out of these words, 140 words are actual misspelled words (247 words are reported as misspelled words while they are correct words). Character *bi-gram* is unable to detect 131 misspelled words of a total of 271 words. This is due to the fact that most of the combination of any two characters in Arabic language is valid word. Character *bi-gram* achieved 51.66%, 36.81% and 42.55% recall, precision and F_1-measure respectively. Table 5.30 shows the spell checking results on CG2-5 data using Character *bi-gram*. Table 5.41 shows the results compared to other methods.

Table 5.30 Spell checking results on CG2-5 data using character Bi-gram

Total misspelled words	271
Total detected	387
Actual misspelled detected	140
Actual misspelled undetected	131
Correct word detected as misspelled	247
Recall	**51.66**
Precision	**36.18**
F_1-measure	**42.55**

- ***Character Tri-gram***

Character *tri-gram* detected 308 words as misspelled words of CG2-5 data. Out of these words, 196 words are actual misspelled words (112 words are reported as misspelled words while they are correct words). Character *tri-gram* is unable to detect 67 misspelled words of a total of 263 words. This is because the most of combination of any three characters in Arabic language is valid word. Character *tri-gram* achieved 74.52%, 63.64% and 68.65% recall, precision and F_1-measure respectively. Table 5.31 shows the spell checking results on CG2-5 data using Character *tri-gram*. Table 5.41 shows the results compared to other methods.

Table 5.31 Spell checking results on CG2-5 data using character Tri-gram

Total misspelled words	263
Total detected	308
Actual misspelled detected	196
Actual misspelled undetected	67
Correct word detected as misspelled	112
Recall	**74.52**
Precision	**63.64**
F_1-measure	**68.65**

- ***Character Quad-gram***

Character *quad-gram* detected 258 words as misspelled words of CG2-5 data. Out of these words, 231 words are actual misspelled words (27 words are reported as misspelled words while they are correct words). Character *quad-gram* is unable to detect 19 misspelled words of a total of 250 words. This is because the large number of combination of any four characters in Arabic language may be a valid word. Character *quad-gram* achieved 92.40%, 89.53% and 90.94% recall, precision and F_1-measure respectively. Table 5.32 shows the spell checking results on CG2-5 data using Character *quad-gram*. Table 5.41 shows the results compared to other methods.

Table 5.32 Spell checking results on CG2-5 data using character Quad-gram

Total misspelled words	250
Total detected	258
Actual misspelled detected	231
Actual misspelled undetected	19
Correct word detected as misspelled	27
Recall	**92.40**
Precision	**89.53**
F_1-measure	**90.94**

- ***Combination of Character n-grams***

The combination of character *n-grams* detected 339 words as misspelled words of CG2-5 data. Out of these words, 241 words are actual misspelled words (98 words are reported as misspelled words while they are correct words). The combination of character *n-grams* is unable to detect 30 misspelled words of a total of 271 words. The combination of character *n-grams* achieved 88.93%, 71.09% and 79.02% recall, precision and F_1-measure respectively. Table 5.33 shows the spell checking results on CG2-5 data using the combination of character *n-grams*. Table 5.41 shows the results compared to other methods.

Table 5.33 Spell checking results on CG2-5 data using the combination of character n-grams

Total misspelled words	271
Total detected	339
Actual misspelled detected	241
Actual misspelled undetected	30
Correct word detected as misspelled	98
Recall	**88.93**
Precision	**71.09**
F_1-measure	**79.02**

- **Computer Generated (CG2-10) data**

  - ***Character Bi-gram***

Character *bi-gram* detected 518 words as misspelled words of CG2-10 data. Out of these words, 277 words are actual misspelled words (241 words are reported as misspelled words while they are correct words). Character *bi-gram* is unable to detect 225 misspelled words of a total of 502 words. This is due to the fact that most of the combination of any two characters in Arabic language is valid word. Character *bi-gram* achieved 55.18%, 53.47% and 54.31% recall, precision and F_1-measure respectively. Table 5.34 shows the spell checking results on CG2-10 data using Character *bi-gram*. Table 5.42 shows the results compared to other methods.

Table 5.34 Spell checking results on CG2-10 data using character Bi-gram

Total misspelled words	502
Total detected	518
Actual misspelled detected	277
Actual misspelled undetected	225
Correct word detected as misspelled	241
Recall	**55.18**
Precision	**53.47**
F_1-measure	**54.31**

  - ***Character Tri-gram***

Character *tri-gram* detected 475 words as misspelled words of CG2-10 data. Out of these words, 358 words are actual misspelled words (117 words are reported as misspelled words while they are correct words). Character *tri-gram* is unable to detect 120 misspelled words of a total of 478 words. This is because the most of combination of any three characters in Arabic language is valid word. Character *tri-gram* achieved 74.90%, 75.37% and 75.13% recall, precision and F_1-measure respectively. Table 5.35 shows the spell checking results on CG2-10 data using Character *tri-gram*. Table 5.42 shows the results compared to other methods.

Table 5.35 Spell checking results on CG2-10 data using character Tri-gram

Total misspelled words	478
Total detected	475
Actual misspelled detected	358
Actual misspelled undetected	120
Correct word detected as misspelled	117
Recall	**74.90**
Precision	**75.37**
F_1-measure	**75.13**

- ***Character Quad-gram***

Character *quad-gram* detected 424 words as misspelled words of CG2-10 data. Out of these words, 385 words are actual misspelled words (39 words are reported as misspelled words while they are correct words). Character *quad-gram* is unable to detect 47 misspelled words of a total of 432 words. This is because a large amount of combination of any four characters in Arabic language may be a valid word. Character *quad-gram* achieved 89.12%, 90.80% and 89.95% recall, precision and F_1-measure respectively. Table 5.36 shows the spell checking results on CG2-10 data using Character *quad-gram*. Table 5.42 shows the results compared to other methods.

Table 5.36 Spell checking results on CG2-10 data using character Quad-gram

Total misspelled words	432
Total detected	424
Actual misspelled detected	385
Actual misspelled undetected	47
Correct word detected as misspelled	39
Recall	**89.12**
Precision	**90.80**
F_1-measure	**89.95**

- ***Combination of Character n-grams***

The combination of character *n-grams* detected 542 words as misspelled words of CG2-10 data. Out of these words, 428 words are actual misspelled words (114 words are reported as misspelled words while they are correct words). The combination of character *n-grams* is unable to detect 74 misspelled words of a total of 502 words. The combination of character *n-grams* achieved 85.26%, 78.97% and 81.99% recall, precision and F_1-measure respectively. Table 5.37 shows the spell checking results on CG2-10 data using the combination of character *n-grams*. Table 5.42 shows the results compared to other methods.

Table 5.37 Spell checking results on CG2-10 data using the combination of character n-grams

Total misspelled words	502
Total detected	542
Actual misspelled detected	428
Actual misspelled undetected	74
Correct word detected as misspelled	114
Recall	**85.26**
Precision	**78.97**
F_1-measure	**81.99**

5.5. SUMMARY

Tables 5.38 to Table 5.42 summarize the methods that we used to spell checking. The best method is the one which combine the Buckwalter's Arabic morphological analyzer (BAMA) and the dictionary look-up in terms of F_1-measure (ATR, CG1-5, CG1-10, CG2-5 and CG2-10) data sets. So, we use the output of the combination to evaluate our spell correction in the next chapter.

Table 5.38 Spell checking results on Arabic Text Recognition (ATR) data

Misspelled Words	**BAMA**	**Dic. Look Up**	**Dic. Look-Up & BAMA**	**Ch. Bi-gram**	**Ch. Tri-gram**	**Ch. Quad-gram**	**Combination of Ch. n-grams**
Total misspelled words	345	345	345	345	339	322	345
Total detected	466	455	347	202	159	266	292
Actual misspelled detected	342	343	340	21	113	215	221
Actual misspelled undetected	3	2	5	324	226	107	124
Correct word detected as misspelled	124	112	7	181	46	51	71
Recall	**99.13**	**99.42**	**98.55**	**6.09**	**33.33**	**66.77**	**64.06**
Precision	**73.39**	**75.38**	**97.98**	**10.40**	**71.07**	**80.83**	**75.68**
F_1-measure	**84.34**	**85.75**	**98.27**	**7.68**	**45.38**	**73.13**	**69.39**

Table 5.39 Spell checking results on CG1-5 data

Misspelled Words	**BAMA**	**Dic. Look Up**	**Dic. Look-Up & BAMA**	**Ch. Bi-gram**	**Ch. Tri-gram**	**Ch. Quad-gram**	**Combination of Ch. n-grams**
Total misspelled words	282	282	282	282	270	243	282
Total detected	316	320	281	345	245	210	210
Actual misspelled detected	280	281	279	94	136	185	185
Actual misspelled undetected	2	1	3	188	134	58	97
Correct word detected as misspelled	36	39	2	251	109	25	25
Recall	**99.29**	**99.65**	**98.94**	**33.33**	**50.37**	**76.13**	**65.60**
Precision	**88.61**	**87.81**	**99.29**	**27.25**	**55.51**	**88.10**	**88.10**
F_1-measure	**93.65**	**93.36**	**99.11**	**29.98**	**52.82**	**81.68**	**75.20**

Table 5.40 Spell Checking results on CG1-10 data

Misspelled Words	**BAMA**	**Dic. Look Up**	**Dic. Look-Up & BAMA**	**Ch. Bi-gram**	**Ch. Tri-gram**	**Ch. Quad-gram**	**Combination of Ch. n-grams**
Total Misspelled Words	545	545	545	545	528	490	545
Total Detected	576	591	549	423	396	402	510
Actual Misspelled Detected	544	544	543	189	285	373	411
Actual Misspelled Undetected	1	1	2	356	243	117	134
Correct word Detected as Misspelled	32	47	6	234	111	29	99
Recall	**99.82**	**99.82**	**99.63**	**34.68**	**53.98**	**76.12**	**75.41**
Precision	**94.44**	**92.05**	**98.91**	**44.68**	**71.97**	**92.79**	**80.59**
F_1-measure	**97.06**	**95.77**	**99.27**	**39.05**	**61.69**	**83.63**	**77.91**

Table 5.41 Spell Checking results on CG2-5 data

Misspelled Words	**BAMA**	**Dic. Look Up**	**Dic. Look-Up & BAMA**	**Ch. Bi-gram**	**Ch. Tri-gram**	**Ch. Quad-gram**	**Combination of Ch. n-grams**
Total misspelled words	271	271	271	271	263	250	271
Total detected	307	314	272	387	308	258	339
Actual misspelled detected	271	270	270	140	196	231	241
Actual misspelled undetected	0	1	1	131	67	19	30
Correct word detected as misspelled	36	44	2	247	112	27	98
Recall	**100**	**99.63**	**99.63**	**51.66**	**74.52**	**92.40**	**88.93**
Precision	**88.27**	**85.99**	**99.26**	**36.18**	**63.64**	**89.53**	**71.09**
F_1-measure	**93.77**	**92.31**	**99.45**	**42.55**	**68.65**	**90.94**	**79.02**

Table 5.42 Spell Checking results on CG2-10 data

Misspelled Words	**BAMA**	**Dic. Look Up**	**Dic. Look-Up & BAMA**	**Ch. Bi-gram**	**Ch. Tri-gram**	**Ch. Quad-gram**	**Combination of Ch. n-grams**
Total misspelled words	502	502	502	502	478	432	502
Total detected	538	586	505	518	475	424	542
Actual misspelled detected	500	501	499	277	358	385	428
Actual misspelled undetected	2	1	3	225	120	47	74
Correct word detected as misspelled	38	85	6	241	117	39	114
Recall	**99.60**	**99.80**	**99.40**	**55.18**	**74.90**	**89.12**	**85.26**
Precision	**92.94**	**85.49**	**98.81**	**53.47**	**75.37**	**90.80**	**78.97**
F_1-measure	**96.15**	**92.10**	**99.11**	**54.31**	**75.13**	**89.95**	**81.99**

CHAPTER 6

ARABIC SPELL CORRECTION PROTOTYPE

Arabic spell correction corrects the errors that are present in the input text. Our Arabic spell correction prototype has the following steps for each erroneous word:

1. Generate a list of candidate words;
2. Rank the candidate words;
3. Correct the erroneous words.

Several approaches are used that are based on minimum edit distance, similarity key, character and word N-grams, and probabilities that are proposed to accomplish these steps (Golding 1995) (Kukich 1992) (Kemighan et al. 1990) (Elmi & Evens 1998). Minimum edit distance is the most popular one to date (Verberne 2002).

6.1. GENERATING CANDIDATE WORDS

This step is used to generate suggested words from uni-gram dictionary that are similar to misspelled word. The words' candidates are formed by adding, deleting or replacing 1 or 2 letters to/from the input word. The used technique in this phase is the minimum edit distance technique.

After detecting the misspelled word, we use edit distance technique, Damerau-Levenshtein distance (Damerau 1964), to generate the best candidate words for the misspelled word. Damerau-Levenshtein algorithm measures the distance between two words by computing the minimum number of editing operations (insertions, deletions, and substitutions) required to change one word into another. Figure 6.1 shows the flowchart for generating candidate words based on the minimum edit distance. The edit distance algorithm is summarized in Algorithm 6.1. The algorithm proceeds by dynamically filling an *m* x *n* matrix where *m* and *n* are the sizes of the compared words. Initially, a cost of 0 is assigned to the matching letters and 1 otherwise. Then, the cells are filled by checking the cost of the three neighboring cells (left, above and diagonally to the left) and adding the cost of the operation in the cell taking the minimum of the three cells. Finally, the distance is read from the cell at the bottom most right of the array. Figure 6.2 shows the result of applying the algorithm to the distance between two words "**مطبعة**"' and "**مطببعة**".

After collecting the list of candidates, we assign weights to the candidates based on their edit distance from the input word. We tried many values of edit distance one and two. We found, experimentally, the optimal values are 0.6 and 0.4 respectively. So, a weight of 0.6 is assigned to all the words with edit distance of 1 and 0.4 to the words with edit distance of 2. Those weights are used to favor the words with the single and double errors.

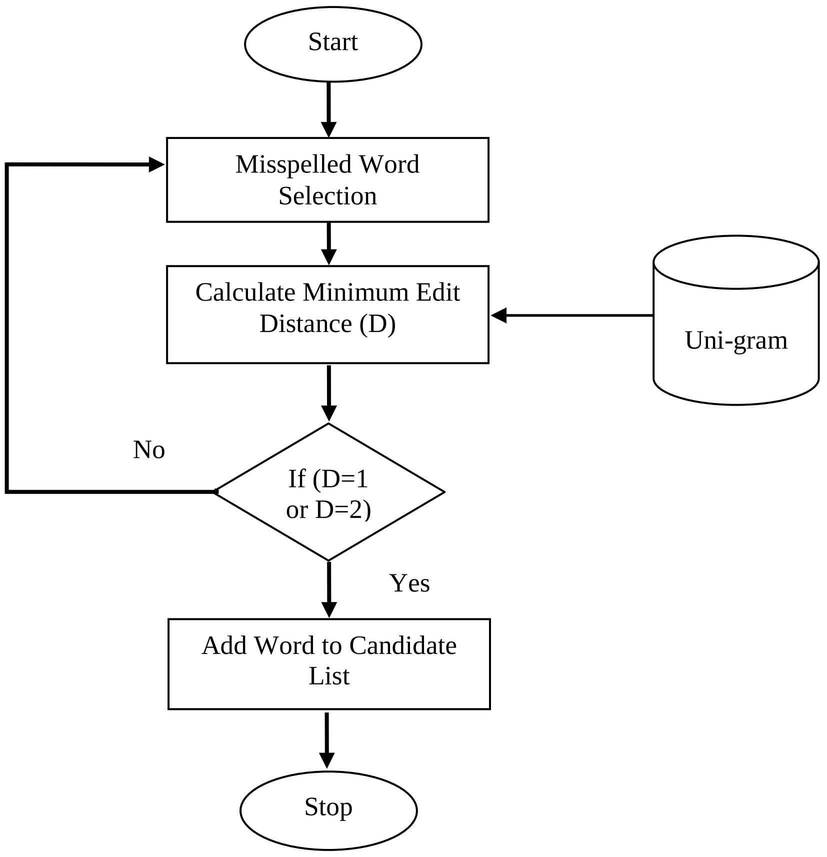

Start
Misspelled Word Selection
Calculate Minimum Edit Distance (D)
Uni-gram
No
If (D=1 or D=2)
Yes
Add Word to Candidate List
Stop

Figure 6.1 Flowchart for generating candidate words based on the minimum edit distance

```
Input: Two words ( str1 and str2)
Output:  Minimum Edit Distance (ED)

for i from 1 to length(str1)
        for j from 1 to length(str2)
             if str1[i] = str2[j]  then
                cost := 0
                  else
                 cost := 1
             end if
            ED [i, j] := minimum( ED [i-1,  j  ] + 1,     // deletion
                                  ED [i  ,  j-1] + 1,     // insertion
                                  ED [i-1,  j-1] + cost   // substitution   )
        end for
end for
```

Algorithm 6.1 The minimum edit distance algorithm

		م	ط	ب	ب	ع	ة
	0	1	2	3	4	5	6
م	1	0	1	2	3	4	5
ط	2	0	0	0	0	0	0
ب	3	0	0	0	0	0	0
ع	4	0	0	0	0	0	0
ة	5	0	0	0	0	0	0

i=1

		م	ط	ب	ب	ع	ة
	0	1	2	3	4	5	6
م	1	0	1	2	3	4	5
ط	2	1	0	1	2	3	4
ب	3	0	0	0	0	0	0
ع	4	0	0	0	0	0	0
ة	5	0	0	0	0	0	0

i=2

		م	ط	ب	ب	ع	ة
	0	1	2	3	4	5	6
م	1	0	1	2	3	4	5
ط	2	1	0	1	2	3	4
ب	3	2	1	0	1	2	3
ع	4	0	0	0	0	0	0
ة	5	0	0	0	0	0	0

i=3

		م	ط	ب	ب	ع	ة
	0	1	2	3	4	5	6
م	1	0	1	2	3	4	5
ط	2	1	0	1	2	3	4
ب	3	2	1	0	1	2	3
ع	4	3	2	1	1	1	2
ة	5	0	0	0	0	0	0

i=4

		م	ط	ب	ب	ع	ة
	0	1	2	3	4	5	6
م	1	0	1	2	3	4	5
ط	2	1	0	1	2	3	4
ب	3	2	1	0	1	2	3
ع	4	3	2	1	1	1	2
ة	5	4	3	2	2	2	1

i=5

Figure 6.2 Illustration of the operation of Damerau –Levenshtein algorithm for finding edit distance between the words "مطبعة" and " مطببعة"

Example: Suppose the sentence " **تطوير آفاق التعاون بين الجامعة والمجنمع** ". The word "**والمجنمع** " is a misspelled word. The candidate words for this misspelled word are shown in the Table 6.1

Table 6.1 List of candidate words from the misspelled word "والمجنمع"

والمجتمعة	والمجامع
والمجتمعي	والمجتمع
والمجرم	والمجمع
والمجزع	المجامع
والمجمد	المجتمع
والمجمر	المجمع
والمجمعة	بالمجامع
والمجموع	بالمجتمع
والمجنون	بالمجمع
والمجني	فالمجامع
والمزمع	فالمجتمع
والمستمع	فالمجمع
والمصنع	كالمجتمع
والمصنوع	والتجمع
والمطامع	والجامع
والممنوع	والجمع
والمنبع	والمانع
والمنع	والمتنوع
وبالمجتمع	والمجامر
وللمجتمع	والمجاميع

6.2. Ranking Candidate Words

This step is used to rank the candidate words in a descending order according to their probabilities. Language models (word n-grams) are used in this phase. After finding the candidate words and assigning the weights, the probability of the tri-gram, bi-gram and uni-gram of each candidate word that are collected from our corpus are used. To do that: we replace the misspelled word with the candidate word, and extract n-grams containing the candidate word. After that, we sort all n-grams (tri-grams, bi-grams and uni-grams) extracted according to their probabilities from highest to lowest.

Note that the probability here indicates the tri-gram probability multiplied by the weight if the tri-gram exists, the bigram probability multiplied by the weight if the bigram exists and the unigram probability multiplied by the weight otherwise as follows:

$$P_i = \alpha * P(W_i \mid W_{i-1}, W_{i-2}) \qquad \text{if } P(W_i \mid W_{i-1}, W_{i-2}) \mathrel{!=} 0$$

$$P_i = \alpha * P(W_i \mid W_{i-1}) \qquad \text{if } P(W_i \mid W_{i-1}, W_{i-2}) = 0 \text{ and } P(W_i \mid W_{i-1}) \mathrel{!=} 0$$

$$P_i = \alpha * P(W_i) \qquad \text{if } P(W_i \mid W_{i-1}, W_{i-2}) = 0 \text{ and } P(W_i \mid W_{i-1}) = 0$$

Where “ i ” represents the candidate number in the list and “i-1” is the token preceding the input word in the original text and “α” is the weight assigned to candidate i. The following procedure is used to rank the candidate words:

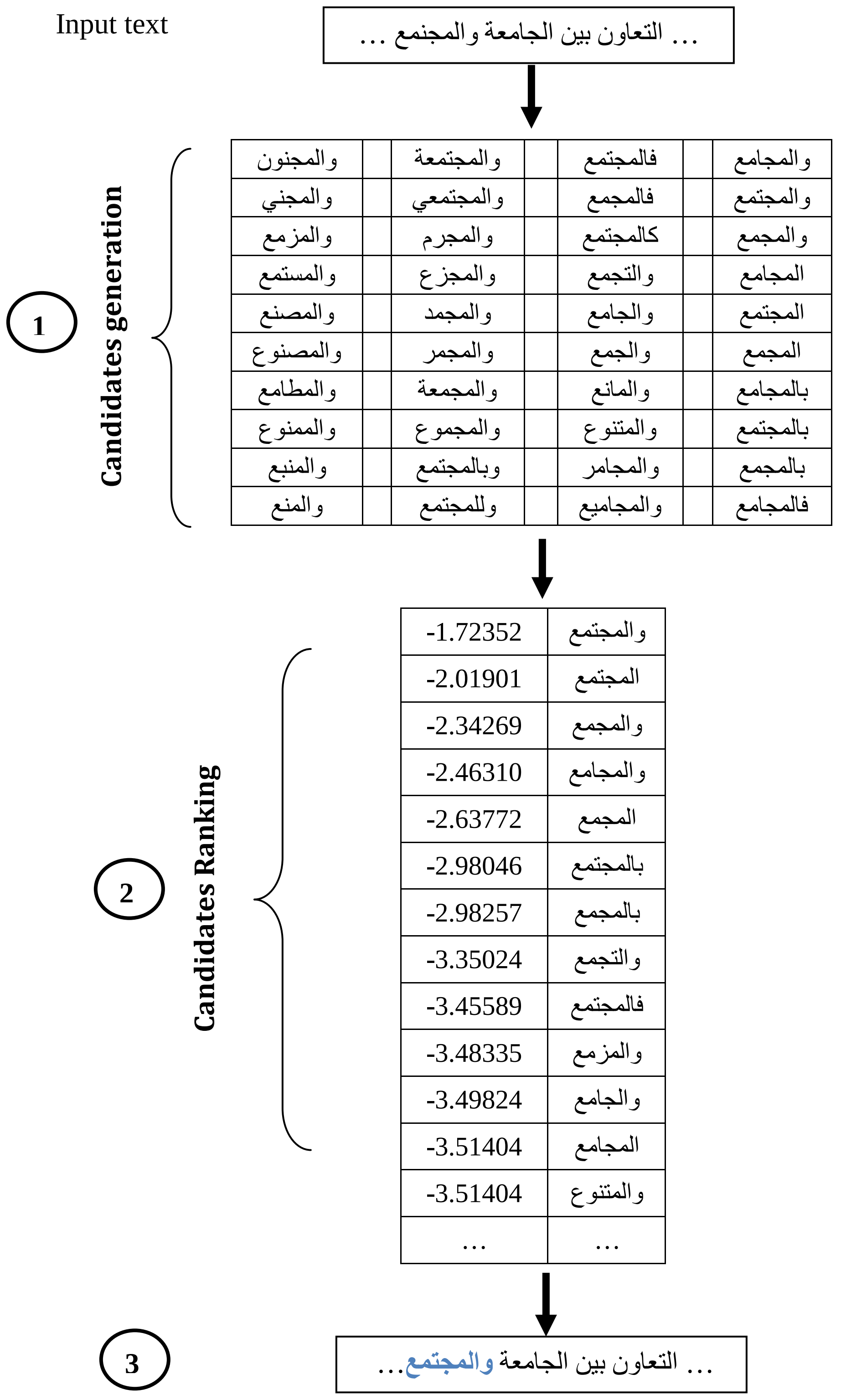
```
Procedure RankingCandidateWords

For each of the candidate words
        Begin
          - Extract tri-grams containing the candidate words and re-sort based on their
            probability.
          - If tri-grams do not exist or number of extracted tri-grams less than 10 then
                -  Extract bi-grams containing the candidate words and re-sort based on
                   their probability.
                -  If bi-grams do not exist or number of extracted bi-grams less than 10
                   then
                      - Perform uni-grams and re-sort based on their probability
                   End
            End
    End
```

6.3. CORRECTING ERRONEOUS WORDS

After ranking all candidate words, this step is used to select the best N candidate word that has the highest rank. To do that we filter the list of candidate words to contain at most N candidates ordered according to their probabilities from highest to lowest. In the case of automatic spell checking and correction, the highest-ranked candidate word is selected as the correct word.

An example for spelling correction process is shown in Figure 6.2. The process consists of three steps. In Step 1, edit distance technique is used to generate all candidate words that are similar to the misspelled word "والمجنمع". As we can see on the Figure 6.2, there are 40 candidate words. In Step 2, word n-grams, tri-grams, bi-grams and uni-grams is used to rank all those candidate words. In Step 3, the best candidate word "والمجتمع" is selected as the correct word based on the highest ranked candidate word.


Input text
... التعاون بين الجامعة والمجنمع ...
1
Candidates generation
2
Candidates Ranking
3
... التعاون بين الجامعة والمجتمع...

والمجنون		والمجتمعة		فالمجتمع		والمجامع
والمجني		والمجتمعي		فالمجمع		والمجتمع
والمزمع		والمجرم		كالمجتمع		والمجمع
والمستمع		والمجزع		والتجمع		المجامع
والمصنع		والمجمد		والجامع		المجتمع
والمصنوع		والمجمر		والجمع		المجمع
والمطامع		والمجمعة		والمانع		بالمجامع
والممنوع		والمجموع		والمتنوع		بالمجتمع
والمنبع		وبالمجتمع		والمجامر		بالمجمع
والمنع		وللمجتمع		والمجاميع		فالمجامع

-1.72352	والمجتمع
-2.01901	المجتمع
-2.34269	والمجمع
-2.46310	والمجامع
-2.63772	المجمع
-2.98046	بالمجتمع
-2.98257	بالمجمع
-3.35024	والتجمع
-3.45589	فالمجتمع
-3.48335	والمزمع
-3.49824	والجامع
-3.51404	المجامع
-3.51404	والمتنوع
…	…

Figure 6.2 Illustration of the spell correction process

6.4. Arabic Spell Correction using OCR Confusion Matrix

In the spell correction of this section, we use a confusion matrix with word n-grams to correct OCR errors. A confusion matrix is a table that has the statistical information about the counts of recognized characters including the counts of correct, misclassified, deleted, inserted and substituted characters etc. The confusion matrix is generated from an OCR system based on the recognition of scanned textual documents. Figure 6.3 shows an example of a confusion matrix.

Arabic spell correction for OCR data is done in the following three steps for word errors:

1. Generate a list of candidate words;
2. Rank the candidate words;
3. Correct the erroneous words.

Arabic spell correction using OCR confusion matrix tries to tackle the three types of OCR errors viz. substitution, insertion, and deletion.

After analyzing the OCR errors on Arabic Text Recognition (ATR) data, we found that the OCR errors occur in the following order:

1- Substitution one character by another

2- Deletion one character

3- Insertion one character

4- Substitution of one character followed by inserting one character

5- Substitution of one character followed by deleting one character

6- Substitution of one character followed by substituting one character

6.4.1. Generating Candidate Words

After detecting the OCR misspelled word, we use character confusion matrix with uni-gram dictionary to generate the best candidate words for the misspelled word. To do that, we apply an iterative process starting from the first character to the last character of the word through three possible edits on characters: substitution, insertion and deletion. Generate candidate words using the OCR confusion matrix algorithm shows in Algorithm 6.2, Algorithm 6.3, Algorithm 6.4 and Algorithm 6.5.

The procedure that we use in the case of insertion is as follows: first, find the character that has the highest insertion counts in confusion matrix of the word. Then, delete it and generate a new word. After that, check the new word in our dictionary. If the new word is found, then it is added to the candidate list. This process is repeated for all the characters of word starting from the character with the highest counts of insertion to the character with the lowest counts as shown in Algorithm 6.3.

In the case of deletion, the procedure is as follows: first, find the character that has highest deletion counts of the confusion matrix. Then, insert it in the different positions of the word, starting from the first position. After that, all generated words are checked in our dictionary. All valid words are added into the candidate list. This process is repeated for all characters deletion counts starting from the character with the highest deletion counts to the character with the lowest deletion counts as shown in Algorithm 6.4.

In the case of substitution, the procedure is as follows: For each character of the word, get all possible substitution characters from the confusion matrix. Then, substitute it with corresponding characters starting from the character with the highest counts of substitution to the character with the lowest counts. Next, check if the resulting word is in our dictionary. If the word is found in the dictionary, then it is added to the candidate list. This process is repeated starting from the first character to the last character of the word as shown in Algorithm 6.5.

```
Input:    Misspelledword  /* error word
Output:  LCW               /*list of candidate words
Begin
   1.  Generate the candidate words by substitution;
   2.  Generate the candidate words by insertion;
   3.  Generate the candidate words by deletion;
   4.  If no words are generated in steps 1 to 3 or the number of generated words
       are less than 10
          5.  Generate the candidate words by substituting one character followed
              by inserting one character;
          6.  If no words are generated in step 5 or the number of generated words
              are less than10
                 7.  Generate the candidate words by substituting one character
                     followed by deleting one character;
                   8.  If no words are generated in step 7 or the number of generated
                       words are less than10
                       9.  Generate the candidate words by substituting one character
                           followed by substituting another character;

End
```

Algorithm 6.2 Generate candidate words using the OCR confusion matrix

```
Input:   Misspelledword  */ error word
Output: $LCW$   /*list of candidate words
Begin
        Get the insertion counts for each character of the MisspelledWord;
        For each character $Ch$ in MisspelledWord
        Begin
            NewWord= Delete $Ch$  that has highest insertion counts;
            If dictionary contains (NewWord) then
                Add NewWord into $LCW$ ;
        End
End
```

Algorithm 6.3 Generate candidate words by insertion

```
Input:   Misspelledword  */ error word
Output: $LCW$   /* list of candidate words
Begin
   Get all characters that have deletion counts of the confusion matrix.
   For i from 1 to count of characters deletion
      Begin
            NewWord=Insert the character that has highest deletion counts;
            For each $W$  in NewWords
            Begin
               If dictionary contains ($W$) then
                   Add $W$ into $LCW$ ;
              End
      End
End
```

Algorithm 6.4 Generate candidate words by deletion

```
Input:   Misspelledword  */ error word
Output: LCW   /*list of candidate words
Begin
        Get all possible substitution characters;
        For each character Ch in MisspelledWord
        Begin
            NewWords = Substitute Ch with its corresponding characters starting
                            from the highest substitution counts;
            For each W  in NewWords
            Begin
                If dictionary contains (W) then
                    Add W into LCW ;
            End
        End
End
```

Algorithm 6.5 Generate candidate words by substitution

The following examples show how to generate candidate words using the OCR confusion matrix.

Example 1: Generate candidate words by insertion.

Let the word, '**ختصر**', be a misspelled word.

Step 1: Get the insertion counts for each character from the confusion matrix that is illustrated in the Figure 6.3. Table 6.2 shows the characters insertion counts.

Table 6.2 The characters insertion counts

Counts	Char
4	خ
9	ت
4	ص
2	ر

Step 2: Delete the character that has highest insertion counts.

Character "ت" has the highest insertion counts. Then, "ت" will be deleted and the new word is "خصر".

Step3: Check the new word in the dictionary.

The new word "خصر" is a valid word. As a result, it is added into the candidate list. Repeat steps 2 and 3 until it exceeds the length of the word. As a result, we added the following candidate words {"خصر", "تصر"} in the candidate list because they are found in the dictionary. However, the other candidate words {"ختر"," ختص"} will be ignored because they are not found in the dictionary.

Example 2: Generate candidate words by deletion

Let the word "ختصر" be a misspelled word.

Step 1: Get all characters that have deletion counts of the confusion matrix. Table 6.3 shows the characters deletion counts.

Table 6.3 The characters deletion counts

ح	ب	ث	ي	ف	م	ا	ن	ر	ك	خ	ت	ل	أ	و	ق	د
15	8	4	11	5	24	67	5	5	5	2	12	14	1	7	2	5
ع	ى	س	ئ	ه	ذ	ج	ش	ض	ظ	ط						
24	1	12	2	2	2	2	3	4	1	1						

Step 2: Insert the character that has highest deletion counts.

Here character "ا" has the highest deletion count. Then, "ا" is inserted in different positions of the misspelled word "ختصر" as shown in the following list {"اختصر", "خاتصر", "ختاصر", "ختصار", "ختصرا"}.

Step 3: Check the new words generated in step 2 in the dictionary.

The new word "اختصر" is the only valid word. As a result, it is added to the candidate list.

Repeat steps 2 and 3 for all characters having deletion counts or the count of the valid candidate words exceed ten. As a result, we added the following candidate words { "اختصر" , "مختصر", "تختصر" , "يختصر" , "نختصر", " أختصر"} in the candidate list because they are found in the dictionary. However, the other generated words will be ignored because they are not found in the dictionary.

Example 3: Generate candidate words by substitution

Let the word "ختصر" is a misspelled word.

Step 1: Get all possible substitution characters for each character of the word from the confusion matrix. Table 6.4 shows the characters substitution counts.

Table 6.4 The characters substitution counts

ر		ص		ت		خ	
1	ب	1	ح	2	ث	16	ح
1	ز	15	م	1	ي	2	ع
22	و	1	و	1	ف	1	ه
2	د	4	ع	15	ن	3	غ
1	ض	6	س	1	ل		
		1	ه	1	س		
		1	ج				

Step 2: Substitute the first character with its corresponding characters starting from the character with the highest count to the character with the lowest count. Here character "ح" has the highest substitution count. Then, "خ" is substituted with "ح" of the

misspelled word "ختصر". As a result, the generated word is "حتصر". This step is repeated for all characters in the substitution character list of "خ". As a result, we get the following list {"حتصر", "غتصر ", "عتصر ", "هتصر"}.

Step 3: Check the new words generated in step 2 in the dictionary.

Repeat steps 2 and 3 for all characters or the count of the valid candidate words exceed ten. As a result, we add the candidate words {"خنصر"} in the candidate list because it is found in the dictionary. However, the other generated words will be ignored because they are not found in the dictionary.

	ح	ب	ث	ي	ة		ف	م	ا	ن	ر	ك	ز	خ	ت	ل	أ	و	ق	د	ع	ى	س	ئ	ه	ذ	ص	ج	.	إ	ش	:	ط	غ	ض	ؤ	ء	آ	ظ	Del	HMM	Truth	
ح	222	0	0	0	0	0	0	8	1	0	0	0	0	16	0	0	0	1	0	1	3	0	1	0	0	0	1	8	0	0	0	0	0	2	1	0	0	0	0	15	280	280	
ب	0	580	0	3	0	0	0	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	596	596	
ث	0	2	98	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	2	0	0	1	0	0	0	0	0	4	111	111	
ي	0	7	0	987	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	11	1011	1011	
ة	0	0	0	0	293	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	299	301	
	0	0	0	0	0	2981	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	2994	2994	
ف	0	0	0	0	0	0	466	1	0	0	0	0	0	0	1	1	0	0	8	3	0	0	0	0	3	0	0	0	0	0	0	0	0	5	0	0	1	0	0	5	494	494	
م	3	0	0	0	0	0	0	671	1	0	0	0	0	0	0	1	0	0	0	0	3	0	7	0	1	0	15	0	0	0	0	0	1	0	0	0	0	0	0	24	727	727	
ا	0	0	0	0	0	1	0	0	1858	0	0	0	0	0	0	1	13	0	0	0	0	0	0	0	1	0	0	0	0	5	0	0	0	0	0	0	0	0	0	67	1946	1946	
ن	1	0	1	0	0	0	0	0	0	668	0	0	0	0	15	4	0	0	3	3	1	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	5	705	705	
ر	0	0	0	0	0	0	0	0	0	0	694	0	2	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	703	703	
ك	0	1	0	0	0	0	0	0	0	2	0	165	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	5	174	174	
ز	0	0	0	0	0	0	0	0	0	0	1	0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	65	65	
خ	5	0	0	0	0	0	0	1	0	0	0	3	0	145	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	2	160	160	
ت	0	1	1	0	0	0	0	0	0	13	0	0	0	0	325	0	0	0	1	6	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	363	365	
ل	2	0	1	0	0	0	1	0	2	3	0	3	0	0	1	1528	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	14	1557	1557	
أ	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	308	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	329	329	
و	0	0	0	0	0	0	0	1	1	0	22	0	0	0	0	0	0	928	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	7	962	962	
ق	1	0	0	0	0	0	4	0	0	5	0	0	0	0	0	1	0	0	389	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	403	403	
د	0	2	0	0	0	0	0	1	1	1	2	0	0	0	0	0	0	0	0	328	0	0	0	0	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	5	345	345	
ع	4	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	388	0	1	0	0	0	4	1	0	0	0	0	1	11	0	0	0	0	0	24	436	436	
ى	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	92	93	
س	1	2	0	1	0	0	1	11	1	9	0	0	0	0	1	3	0	0	0	0	0	1	330	0	0	0	6	0	0	0	1	0	0	0	2	0	0	0	1	12	383	383	
ئ	0	0	0	0	0	0	0	0	1	4	0	0	0	0	0	1	0	0	1	2	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	29	29	
ه	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	537	0	1	0	0	0	0	0	0	1	0	0	1	0	0	2	549	550	
ذ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	1	200	0	0	0	0	0	0	0	0	0	0	0	0	0	2	211	211	
ص	0	0	0	0	0	0	0	13	0	4	0	0	0	0	0	0	0	0	0	0	1	1	6	0	0	0	111	0	0	0	0	0	0	0	12	0	0	0	0	0	148	148	
ج	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	211	0	0	0	0	0	0	0	0	0	0	0	2	217	217	
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	1	0	0	1	52	52	
إ	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	149	0	0	0	0	0	0	0	0	0	3	155	155	
ش	1	0	3	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	134	0	0	0	0	0	0	0	0	3	147	147	
:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	51	0	0	0	0	0	0	0	0	51	51	
ط	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	144	0	0	0	0	0	2	1	154	154	
غ	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	64	0	0	0	0	1	0	82	82	
ض	0	0	0	0	0	0	0	1	0	2	1	0	0	0	0	1	0	0	0	0	0	1	5	0	0	0	0	0	0	0	4	0	0	1	111	0	0	0	0	4	131	131	
ؤ	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	12	12	
ء	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	73	73	
آ	0	0	0	0	0	0	0	2	3	0	0	0	0	0	0	1	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	32	32	
ظ	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	7	1	15	15	
Ins	12	0	6	3	0	7	1	19	18	31	2	1	1	4	9	22	4	0	1	8	10	1	1	1	12	1	4	1	3	7	3	0	1	2	1	0	10	0	0	0			207
																																							262			

Figure 6.3 OCR confusion matrix

6.4.2. Ranking Candidate Words

This step is used to rank the candidate words in a descending order according to their probabilities. Language models (word n-grams) are used in this phase. After finding the candidate words using the confusion matrix, the probability of the tri-gram, bi-gram and uni-gram of each candidate word that are collected from our corpus are used. The procedure that we use to rank candidate words is as follows: first, replace the misspelled word with each candidate word, and extract the n-grams containing the candidate word. Next, look-up all extracted n-grams in the tri-grams, the bi-grams and the uni-grams. After that, we sort all extracted n-grams (tri-grams, bi-grams and uni-grams) according to their probabilities from highest to lowest.

6.4.3. Correcting Erroneous Words

After ranking all candidate words, this step is used to select the best N candidate words that have the highest-ranks. To do that we filter the list of candidate words to contain at most *N* candidates ordered according to their probabilities from highest to lowest. In the case of automatic spell checking and correction, the highest-ranked candidate word is selected as the correct word.

6.5. EXPERIMENTAL RESULTS

6.5.1. Evaluation Measures

In the previous chapter, we discussed the methods that we used to detect misspelled words (spell checking). It was shown that the best method is the one which combine the Buckwalter’s Arabic morphological analyzer (BAMA) and the dictionary look-up. So, we use the output of the combination to evaluate our spell correction prototype.

In our work, we evaluated spell correction performance using error correction accuracy and precision as a metric.

- **Error Correction Accuracy**

Error correction accuracy is defined as the total number of successful correct suggestions in Top-N candidate suggestions proposed by the prototype over the number of the misspelled detected words by the prototype. Equation 6.1 shows error correction accuracy (Zhang et al. 2007) (G. Huang et al. 2010)

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{Number\ of\ Valid\ Corretions\ in\ TopN}}{\boldsymbol{\#\ of\ Valid\ Corretions + \#\ ofNo\ Corretions + \#\ of\ Bad\ Corretions}}\ \boldsymbol{x\ 100} \qquad (\boldsymbol{6.1})$$

Where:

- *# of Valid Correction*s = total number of successful correct suggestions in Top-*N* candidate suggestions.

- *# of Valid Corrections* + *# of No Corrections* + *# of Bad Corrections*= total number of the misspelled words detected.
- N is taken as 1, 5, 10

- **Precision**

Precision is defined as the total number of successful correct suggestions in Top-*N* candidate suggestions proposed by the prototype over the total number of corrections. Equation 6.2 shows the precision (Zhang et al. 2007).

$$\boldsymbol{Precision = \frac{\#\ of\ Valid\ Corretion\ in\ TopN}{\#\ of\ Valid\ Corretions + \#\ of\ Bad\ Corretions}\ x\ 100} \qquad (\mathbf{6.2})$$

Where:

# of Valid Corrections = total number of successful correct suggestions in Top-*N* candidate suggestions.

# of Valid Corrections + *# of Bad Corrections* = total number of corrections.

In order to analyze the error correction accuracy and precision we use three different types of outcomes: (1) whether the correct word was ranked in the Top-1, (2) whether the correct word was ranked in the Top-5, and (3) whether the correct word was ranked in the Top-10 candidate suggestions. In addition, we counted words that were corrected, but the correction was bad, and words that were not corrected at all.

6.5.2. Spell Correction Performance

In this section, we present the results for spell correction with respect to the aforementioned evaluation measures.

6.5.2.1. Arabic Text Recognition (ATR) data

- **Spell Correction using OCR Confusion Matrix**

Our spell correction prototype using OCR confusion matrix suggested 332 words as corrected words. Out of these words, 230 words were found in the Top-1 candidate suggestions, 264 words found in the Top-5 candidate suggestions and 274 words found in the Top-10 candidate suggestions. Our spell correction makes bad suggestions to 102, 68 and 58 words in the Top-1, Top-5 and Top-10 candidate suggestions, respectively. This is because those 102, 68 and 58 words contain more than two errors but when we use the confusion matrix, our spell correction prototype using OCR confusion matrix try to generate candidate words that are close to the misspelled word by one or two character insert, replace and delete. However, these candidate words are not equal to the ground truth words. For example, let the OCR error word { "الصلارة" }, the candidate words are: { "الصورة", "الحارة", "الحضارة ", "الصلاة", "الصادرة"} while the actual word is { "الحلاوة" }. In addition, the probability of the candidate word "الحلاوة" is less than other candidate's word as shown in Figure 6.4. So, we can say that those candidate words are not found in Top-1, Top-5 and Top-10 and the correction is a bad or wrong correction. We achieved 66.28%, 76.80% and 78.96% error correction accuracy in the Top-1, Top-5 and Top-10, respectively. In addition, we achieved 69.28%, 79.52% and 82.53% precision in the

Top-1, top5 and Top-10, respectively. Spell correction using OCR confusion matrix is unable to give 15 words any suggestion of a total of 347 words. This is because those 15 words have more than two errors while our spell correction prototype using OCR confusion matrix check only two errors in a word. Figure 6.5 shows a list of un-corrected OCR errors with their corresponding ground truth. Table 6.5 shows the spell correction results on ATR data using confusion matrix.

Candidate word	**Probability**
الصورة	-4.04214
الحارة	-4.23737
الحضارة	-4.26775
الصلاة	-4.31034
الصادرة	-4.33821
السفارة	-4.45987
الولادة	-4.70101
الولاة	-4.79839
المهارة	-5.08312
للصلاة	-5.30497
الحلاوة	**-5.36187**

Figure 6.4 An example of the probability of the candidate word "الحلاوة" is less than other candidate's word

Ground Truth	OCR Errors
أرفارياوس	أرقاريارسل
القنطس	القتغطمل
القبض	القبعنس
ديسقوريدوس	ديسقرزغرصل
الحادث	لخادنثا
الامتزاج	لأستزاج
الإسهال	الإمهمالا
الجحوظ	الجحوخلا
ثلاث	ثلانشب
لبياض	لبياصنس
ديسقوريدوس	ديسقوريشرمن
خبث	حكبنلب
الرصاص	الرتصانمس
الاسفنج	ألأسنفج
وحينئذ	وحيدثنذ

Figure 6.5 List of un-corrected OCR errors using OCR confusion matrix

It is important to note that the OCR recognition rate is less than 90%, which may results in words having more than two errors while we are only checking one or two errors in a word.

Table 6.5 Spell Correction Results on ATR Data using OCR Confusion Matrix

Total Words	3229		
	Top-1	**Top-5**	**Top-10**
Total word detected	347		
Total word corrected	332		
Valid corrections	230	264	274
Bad corrections	102	68	58
Not corrected	15	15	15
Error correction accuracy	**66.28**	**76.08**	**78.96**
Precision	**69.28**	**79.52**	**82.53**

- **Spell Correction using Edit Distance**

Our spell correction prototype using edit distance suggested 338 words as corrected words. Out of these words, 243 words were found in the Top-1 candidate suggestions, 284 words found in the Top-5 candidate suggestions and 293 words were found in the Top-10 candidate suggestions. Our spell correction prototype is unable to suggest 95, 54 and 45 words correctly in the Top-1, Top-5 and Top-10 candidate suggestions, respectively. We achieved 70.03%, 81.84% and 84.44% error correction accuracy in the Top-1, Top-5 and Top-10 respectively. In addition, we achieved 71.89%, 84.02% and 86.61 precision in the Top-1, Top-5 and Top-10, respectively. Spell correction using edit distance is unable to give 9 words any suggestion of a total of 347 words. This is because those 9 words contain more than two errors while we use edit distance to check one or two errors in a word. Figure 6.6 shows a list of un-corrected OCR errors using edit distance algorithm with their corresponding ground truth. Table 6.6 shows the spell correction results on ATR data using the edit distance.

Ground Truth	OCR Errors
أرفارياوس	أرقاريارسل
القنطس	القتغطمل
ديسقوريدوس	ديسقرزغرصل
الحادث	لخادنثا
الجحوظ	الجحوخلا
ديسقوريدوس	ديسقوريشرمن
خبث	حكبنلب
الرصاص	الرتصانمس
الاسفنج	ألأسنفج

Figure 6.6 List of un-corrected OCR errors using Edit Distance

Table 6.6 Spell correction results on ATR data using Edit Distance

Total Words	3229		
	Top-1	**Top-5**	**Top-10**
Total word detected	347		
Total word corrected	338		
Valid corrections	243	284	293
Bad corrections	95	54	45
Not corrected	9	9	9
Error correction accuracy	**70.03**	**81.84**	**84.44**
Precision	**71.89**	**84.02**	**86.69**

Error correction accuracy on Top-1, Top-5 and Top-10 when we use the edit distance on ATR data is little better than when we use the OCR confusion matrix as shown in Figure 6.7.

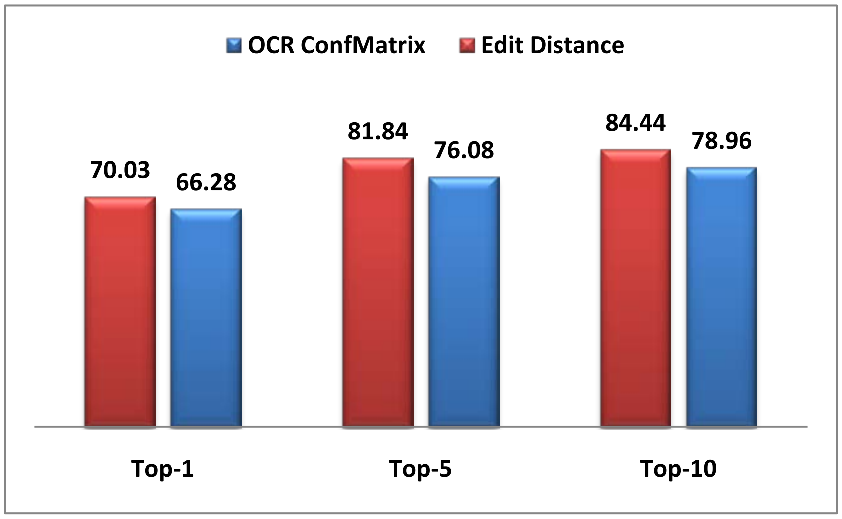


Figure 6.7 Error correction accuracy

6.5.2.2. Computer Generated (CG) data

We used our Computer Generated (CG) data that we discussed in Chapter 4 to test our prototype.

- **CG1-5 data**

Our spell correction prototype suggested 281 words as corrected words. Out of these words, 246 words were found in the Top-1 candidate suggestions, 271 words found in the Top-5 candidate suggestions and 275 words were found in the Top-10 candidate suggestions. Our spell correction prototype is unable to suggest 35, 10 and 6 words correctly in the Top-1, Top-5 and Top-10 candidate suggestions, respectively. We achieved 87.54%, 96.44% and 97.86% error correction accuracy in the Top-1, Top-5 and Top-10 respectively. In addition, we achieved 87.54%, 96.44% and 97.86% precision in the Top-1, Top-5 and Top-10 respectively as all words were corrected. Table 6.7 shows the spell correction results on CG1-5 data.

Table 6.7 Spell correction results on CG1-5 data

Total Words	6665		
	Top-1	**Top-5**	**Top-10**
Total word detected	281		
Total word corrected	281		
Valid corrections	246	271	275
Bad corrections	35	10	6
Not corrected	0	0	0
Error correction accuracy	**87.54**	**96.44**	**97.86**
Precision	**87.54**	**96.44**	**97.86**

- **CG1-10 data**

Our spell correction prototype suggested 549 words as corrected words. Out of these words, 490 words were found in the Top-1 candidate suggestions, 537 words found in the Top-5 candidate suggestions and 543 words were found in the Top-10 candidate suggestions. Our spell correction prototype is unable to correct 59 in Top-1 and suggest 12 and 6 words correctly in the Top-5 and Top-10 candidate suggestions, respectively. We achieved 89.25, 97.81% and 98.91% error correction accuracy in the Top-1, Top-5 and Top-10 respectively. In addition, we achieved 89.25, 97.81% and 98.91% precision in the Top-1, Top-5 and Top-10 respectively as all words were corrected. Table 6.8 shows the spell correction results on CG1-10 data.

Table 6.8 Spell correction results on CG1-10 data

Total Words	6665		
	Top-1	**Top-5**	**Top-10**
Total word detected	549		
Total word corrected	549		
Valid corrections	490	537	543
Bad corrections	59	12	6
Not corrected	0	0	0
Error correction accuracy	**89.25**	**97.81**	**98.91**
Precision	**89.25**	**97.81**	**98.91**

- **CG2-5 data**

Our spell correction prototype suggested 271 words as corrected words. Out of these words, 227 words were found in the Top-1 candidate suggestions, 252 words found in the Top-5 candidate suggestions and 258 words were found in the Top-10 candidate suggestions. Our spell correction prototype is unable to suggest 45, 19 and 13 words correctly in the Top-1, Top-5 and Top-10 candidate suggestions, respectively. We achieved 83.46%, 92.65% and 94.85% error correction accuracy in the Top-1, Top-5 and Top-10 respectively. In addition, we achieved 83.76%, 92.99% and 95.20% precision in the Top-1, Top-5 and Top-10 respectively. Our spell correction prototype is unable to give 1 word, "وإقخصاجئها", any suggestion of a total of 272 words. This is because our dictionary does not contain any words close to this word with edit distant one or two. Table 6.9 shows the spell correction results on CG2-5 data.

Table 6.9 Spell correction results on CG2-5 data

Total Words	6665		
	Top-1	**Top-5**	**Top-10**
Total word detected	272		
Total word corrected	271		
Valid corrections	227	252	258
Bad corrections	45	19	13
Not corrected	1	1	1
Error correction accuracy	**83.46**	**92.65**	**94.85**
Precision	**83.76**	**92.99**	**95.20**

- **CG2-10 data**

Our spell correction prototype suggested 504 words as corrected words. Out of these words, 402 words were found in the Top-1 candidate suggestions, 456 words found in the Top-5 candidate suggestions and 460 words were found in the Top-10 candidate suggestions. Our spell correction prototype is unable to suggest 102, 48 and 44 words correctly in the Top-1, Top-5 and Top-10 candidate suggestions, respectively. We achieved 79.60%, 90.30% and 91.09% error correction accuracy in the Top-1, Top-5 and Top-10 respectively. In addition, we achieved 79.76%, 90.48% and 91.27% precision in the Top-1, Top-5 and Top-10 respectively. Our spell correction is unable to give 1 word, "انجامشون", any suggestion of a total of 505 words. This is because our dictionary does not contain any words close to this word with edit distant one or two. Table 6.10 shows the spell correction results on CG2-10 data.

Table 6.10 Spell correction results on CG2-10 data

Total Words	6665		
	Top1	**Top3**	**Top5**
Total word detected	505		
Total word corrected	504		
Valid corrections	402	456	460
Bad corrections	102	48	44
Not corrected	1	1	1
Error correction accuracy	**79.60**	**90.30**	**91.09**
Precision	**79.76**	**90.48**	**91.27**

CHAPTER 7

CONCLUSION AND FUTURE WORK

This chapter presents a summary of our major contributions in this thesis work. The goal of this research is to design and implement a prototype for spell checking and correction for Arabic text recognition that would be able to detect and correct non-word errors automatically. This chapter also discusses possible future research directions.

Spell checking and correction plays a vital rule in many applications such as, word processing, information retrieval, grammar correction, machine translation, OCR and on-line handwriting recognition. In this thesis, we designed and implemented a prototype for spell checking and correction for Arabic text recognition that would be able to detect and correct non-word errors automatically. We collected Arabic text corpus from different resources and build different language models for spell checking and correction. We evaluated the performance of the proposed spell checking and correction prototype. We used Buckwalter's Arabic morphological analyzer (BAMA), the dictionary look-up and the language model on character level to detect non-word errors. For spell correction, we used edit distance techniques and N-gram language models (word n-grams).

Two types of data sets are used. One set, Arabic Text Recognition (ATR) data, which generated from an OCR system developed at KFUPM. The second set, a Computer

Generated (CG) data with errors, which prepared by taking a normal correct text and randomly introducing three types of errors (*insert, delete and replace*).

The results show that the best method is the one which combines the Buckwalter's Arabic morphological analyzer (BAMA) and the dictionary look-up to detect errors word. For spell correction techniques, combining the OCR confusion matrix with language model (word n-grams) performs well on the Arabic Text Recognition. Also, combining edit distance with word n-grams performs well on the Arabic Text Recognition (ATR) data. However, combining edit distance with word n-grams has good performance on the Computer Generated (CG) data set. It is difficult to evaluate our prototype's performance against others' since there is no generally available test set.

As an extension to this work, real word errors detection and correction may be addressed. In addition, the use of a better data structure may be used to represent the edit distance calculations and look-up dictionary techniques to improve the speed. Furthermore, misspelled words with more than two errors may be considered.

REFERENCES

Agirre, E. et al., 1998. Towards a Single Proposal in Spelling Correction. *Proceedings of the 36th annual meeting on Association for Computational Linguistics*.

Ahmed, F., Luca, E. & Nürnberger, A., 2009. Revised N-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness. *Research journal on computer science and computer engineering with applications*, 40, pp.39-48.

Al-Muhtaseb, H., 2010. Arabic Text Recognition of Printed Manuscripts. *PhD Thesis, School of Computing, Informatics & Media, University of Bradford, UK.*

Al-Sulaiti, L., 2004. Designing and Developing a Corpus of Contemporary Arabic. *Master's thesis, School of Computing in the University of Leeds*.

Angell, R., Freund, G. & Willett, P., 1983. Automatic Spelling Correction Using a Trigram Similarity Measure. *Information Processing and Management*, 19(4), pp.255-261.

Austin, J., 1996. Distributed Associative Memories for High-Speed Symbolic Reasoning. *Fuzzy Sets and System*, 82(2), pp.223-233.

Bandyopadhyay, S., 2000. Detection and Correction of Phonetic Errors with a New Orthographic Dictionary. *14th Pacific Asia Conference on Language, Information and Computation (PACLIC 14), Tokyo, Japan*, pp.15-22.

Bhagat, M., 2007. Spelling Error Pattern Analysis of Punjabi Typed Text. *A Thesis Report Submitted in the partial fulfillment of the requirements for the award of the degree of ME in Software Engineering Computer Science and Engineering Department Thapar Institute of Engine*, pp.1-62.

Bowden, T. & Kiraz, G.A., 1995. A Morphographemic Model for Error Correction in Nonconcatenative Strings. *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, p.7.

Boyd, A., 2009. Pronunciation Modeling in Spelling Correction for Writers of English as a Foreign Language. *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pp.31-36.

Brants, T. et al., 2007. Large Language Models in Machine Translation. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague*, 1(June), pp.858-867.

Brill, E. & Moore, R.C., 2000. An improved error model for noisy channel spelling correction. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics ACL 00*, pp.286-293.

Buckwalter, T., 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. *Linguistic Data Consortium, University of Pennsylvania, LDC Catalog No.: LDC2002L49*.

Chen, S. & Goodman, J., 1998. An Empirical Study of Smoothing Techniques for Language Modeling. *Technical Report TR-10-98, Computer Science Group, Harvard University*.

Church, K. & Gale, W., 1991. Probability Scoring for Spelling Correction. *Statistics and Computing*, 1(2), pp.93-103.

Dalkilic, G. & Cebi, Y., 2009. Turkish Spelling Error Detection and Correction by Using Word N-grams. *Fifth International Conference on Soft Computing with Words and Perceptions in System Analysis Decision and Control (2009)*, pp.1-4.

Damerau, F., 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, 7(3), pp.171-176.

Deorowicz, S. & Ciura, G., 2005. Correcting Spelling Errors by Modelling their Causes. *International Journal of Applied Mathematics and Computer Science*, 15(2), pp.275-285.

Elmi, M. & Evens, M., 1998. Spelling Correction using Context. In *Proceedings of the ThirtySixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*. pp. 360-364.

Federico, M. & Cettolo, M., 2007. Efficient Handling of N-gram Language Models for Statistical Machine Translation. *Proceedings of the Second Workshop on Statistical Machine Translation. Association for Computational Linguistics,* (June), pp.88-95.

Federico, M., Bertoldi, N. & Cettolo, M., 2008. IRST Language Modeling Toolkit-Version 5.20.00 USER MANUAL. , pp.1-8.

Flexner, B., 1983. Random House unabridged dictionary. *(2nd ed.). New York: Random House*.

Fossati, D. & Eugenio, B., 2007. A Mixed Trigrams Approach for Context Sensitive Spell Checking. *Gelbukh, A. (Ed.): CICLing 2007. LNCS, vol. 4394, pp. 623–633. Springer, Heidelberg.*

Golding, A., 1995. A Bayesian Hybrid Method for Context-Sensitive Spelling Correction. *Proceedings of the 3rd Workshop on Very Large Corpora, Boston, MA*, pp.39-53.

Golding, A. & Schabes, Y., 1996. Combining Trigram-based and Feature-based Methods for Context-Sensitive Spelling Correction. *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pp.71-78.

Haddad, B. & Yaseen, M., 2007. Detection and Correction of Non-Words in Arabic: A Hybrid Approach. *International Journal of Computer Processing of Oriental Languages (IJCPOL)*, 20(4), pp.237-257.

Hassan, A., Hassan, H. & Noeman, S., 2008. Language Independent Text Correction using Finite State Automata. *Proceedings of the 2008 International Joint Conference on Natural Language Processing (IJCNLP).*

Hodge, V. & Austin, J., 2002. A comparison of a novel neural spell checker and standard spell checking algorithms. *Pattern Recognition*, 35, pp.2571-2580.

Huang, G. et al., 2010. A Misspelling Intelligent Analysis Approach for Correcting Misspelled Words in English Text. *Journal of Convergence Information Technology*, 5(5), pp.58-71.

Islam, A. & Inkpen, D., 2009a. Real-word spelling correction using Google Web IT 3-grams. *In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 3(August), pp.1241-1249.

Islam, A. & Inkpen, D., 2009b. Real-word spelling correction using Google web 1Tn-gram data set. *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM ’09*.

Islam, A. & Inkpen, D., 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data*, 2(2), pp.1-25.

Jurafsky, D. & Martin, J., 2009. Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing.

Kaur, R. & Bhatia, P., 2010. Design and Implementation of SUDHAAR-Punjabi Spell Checker. *International Journal of Information and Telecommunication Technology IJITT*, 1(1), pp.10-15.

Kemighan, M., Church, K. & Gale, W., 1990. A Spelling Correction Program Based on a Noisy Channel Model. *Proceedings of the 13th conference on Computational linguistics*, 2, pp.205-210.

Kondrak, G., 2005. N-gram Similarity and Distance. *In Proceedings of the Twelfth International Conference on String Processing and Information Retrieval (SPIRE 2005), Buenos Aires, Argentina*, pp.115-126.

Kukich, K., 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4), pp.377-439.

Lehal, G., 2007. Design and Implementation of Punjabi Spell Checker. *International Journal of Systemics, Cybernetics and Informatics*, pp.70-75.

Liang, H., 2008. *Spell checkers and correctors: A unified treatment*. Master's thesis, University of Pretoria, South Africa.

Magdy, W. & Darwish, K., 2006. Arabic OCR Error Correction Using Character Segment Correction, Language Modeling, and Shallow Morphology. *Proceedings of the 2006 conference on empirical methods in natural language processing, Sydney, Australia*, (July), pp.408-414.

Magdy, W. & Darwish, K., 2010. Omni Font OCR Error Correction with Effect on Retrieval. In *Intelligent Systems Design and Applications (ISDA)*. pp. 415-420.

Mahmoud, S. & AlMuhtaseb, H., 2010. Automatic Arabic Optical Text Recognition (AOTR). *Final Report, Research Project # IN060337, KFUPM*.

Melamed, D., 1999. Bitext Maps And Alignment Via Pattern Recognition. *Computational Linguistics*, pp.107-130.

Morris, R. & Cherry, L.L., 1975. Computer Detection of Typographical Errors. *Ieee Transactions On Professional Communication*, PC-18(1), pp.54-56.

Naseem, T., 2004. *A Hybrid Approach for Urdu Spell Checking*. Master's thesis,National University of Computer & Emerging Sciences, Pakistan.

Naseem, T. & Hussain, S., 2007. A novel approach for ranking spelling error corrections for Urdu. *Language Resources and Evaluation*, 41(2), pp.117-128.

Oparin, I., 2008. Language Models for Automatic Speech Recognition Inflectional Languages. *PhD Thesis, University of West Bohemia, Plzen, Czech Republic*.

Raab, M., 2006. Language Model Techniques in Machine Translation. *Master's thesis, Universit¨at Karlsruhe / Carnegie Mellon University*, (October).

Riseman, E. & Hanson, A., 1974. A Contextual Postprocessing System for Error Correction Using Binary n-Grams. *IEEE Transactions on Computers*, 23(5), pp.480-493.

Sasaki, Y., 2007. The Truth of the F-measure. *Teaching, Tutorial materials*, pp.1-5.

Schaback, J. & Li, F., 2007. Multi-Level Feature Extraction for Spelling Correction. *In IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data, Hyderabad, India*, pp.79-86.

Shaalan, K., Allam, A. & Gomah, A., 2003. Towards Automatic Spell Checking for Arabic. *Proceedings of the 4th Conference on Language Engineering Egyptian Society of Language Engineering ELSE'03*, pp.240-247.

Shaalan, K., Aref, R. & Fahmy, A., 2010. An Approach for Analyzing and Correcting Spelling Errors for Non-native Arabic learners. In *the Proceedings of The 7th International Conference on Informatics and Systems, INFOS2010*. Cairo, pp. 53-59.

Stolcke, A., 2002. SRILM — An Extensible Language Modeling Toolkit. *Intl. Conf. on Spoken Language Processing*.

Taghva, K. & Stofsky, E., 2001. OCRSpell: an interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition*, 3(3), pp.125-137.

Turner, M. & Austin, J., 1997. Matching Performance of Binary Correlation Matrix Memories. *Neural Networks*, 10(9), pp.1637-1648.

Verberne, S., 2002. Context-Sensitive Spell Checking Based on Word Trigram Probabilities. *Master's thesis, University of Nijmegen*.

Yannakoudakis, E. & Fawthro, D., 1983. The Rrules of Spelling Errors. *Information Processing & Management*, 19(2), pp.87-99.

Zamora, E., Pollock, J. & Zamora, A., 1981. The use of trigram analysis for spelling error detection. *Information Processing & Management*, 17(6), pp.305-316.

Zhai, C., 2008. Statistical Language Models for Information Retrieval A Critical Review. *Foundations and Trends® in Information Retrieval*, 2(3), pp.137-213.

Zhang, Y. et al., 2007. Discriminative Reranking for Spelling Correction. In *Proceedings of the 20th Pacific Asia Conference on Language Information and Computation*. pp. 64-71.

Zhuang, L. et al., 2004. A Chinese OCR Spelling Check Approach Based on Statistical Language Models. *Proceedings of the IEEE International Conference on System, Man and Cybernetics, Hague, Netherlands*, pp.4727-4732.

Zobel, J., Box, G. & Dart, P., 1996. Phonetic String Matching: Lessons from Information Retrieval. *In Proceedings of SIGIR-96, the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zribi, C., Mejri, H. & Ben Ahmed, M., 2007. Combining Methods for Detecting and Correcting Semantic Hidden Errors in Arabic Texts. *In A. Gelbukh, ed. Lecture Notes in Computer Science, Computational Linguistics and Intelligent Text Processing. Springer Berlin / Heidelberg*, pp.634 - 645.

VITA

- Adnan Abdo Mohammed Mahdi
- Born in Thamar, Yemen on September 03,1979
- Graduated from high school in 1998 with grade 90.12%
- Received a scholarship from the Ministry of Higher Education to study B.Sc. degree at Al-Mustansiriya University in 1999
- Received Bachelor of Science (B.Sc) in Computer Science from ***Al-Mustansiriya University***, *Baghdad*, *Iraq*, 2003, with a GPA of 86.786%
    - Ranked first among 102 students
- Received a scholarship from the Ministry of Higher Education and ***Al- Hodeidah University*** to study M.Sc. degree at KFUPM in 2007

- **Contact Details:**
- **Present Address:** Department of Information and Computer Science, ***King Fahd University of Petroleum and Minerals (KFUPM),*** Dhahran, Saudi Arabia.
- **Email Address**: g200704210@kfupm.edu.sa
- **Permanent Address**: Department of Computer Science, ***Al-Hodeidah University,*** Al- Hodeidah, Yemen.
- **Permanent Email Address**: adnannet2004@yahoo.com
  adnanmhdi@gmail.com